பேராலயமும் சந்தையும்

ஒரு தற்செயலான புரட்சியாளரின் லினக்ஸ் மற்றும் திறந்த மூலம் பற்றிய சிந்தனைகள் — பதிப்பு 3.0



எரிக் ரேமண்ட்

தமிழாக்கம்: இரா. அசோகன்

பேராலயமும் சந்தையும்

எரிக் ரேமண்ட்

FreeTamilEbooks.com

பேராலயமும் சந்தையும்

பேராலயமும் சந்தையும்

பேராலயமும் சந்தையும்

எரிக் ரேமண்ட்

மின்னூல் வெளியீடு : FreeTamilEbooks.com

அட்டைப்படம் - லெனின் குருசாமி - guruleninn@gmail.com

மின்னூலாக்கம் - ஐஸ்வர்யா லெனின் - aishushanmugam09@gmail.com

This book was produced using pandoc

பதிவிறக்கம் செய்ய - http://FreeTamilEbooks.com/ebooks/the_cathedral_and_the_bazaar

மின்னூல் வெளியீட்டாளர்: http://freetamilebooks.com
அட்டைப்படம்: லெனின் குருசாமி - guruleninn@gmail.com
மின்னூலாக்கம்: ஐஸ்வர்யா லெனின் - aishushanmugam09@gmail.com
மின்னூலாக்க செயற்திட்டம்: கணியம் அறக்கட்டளை - kaniyam.com/foundation

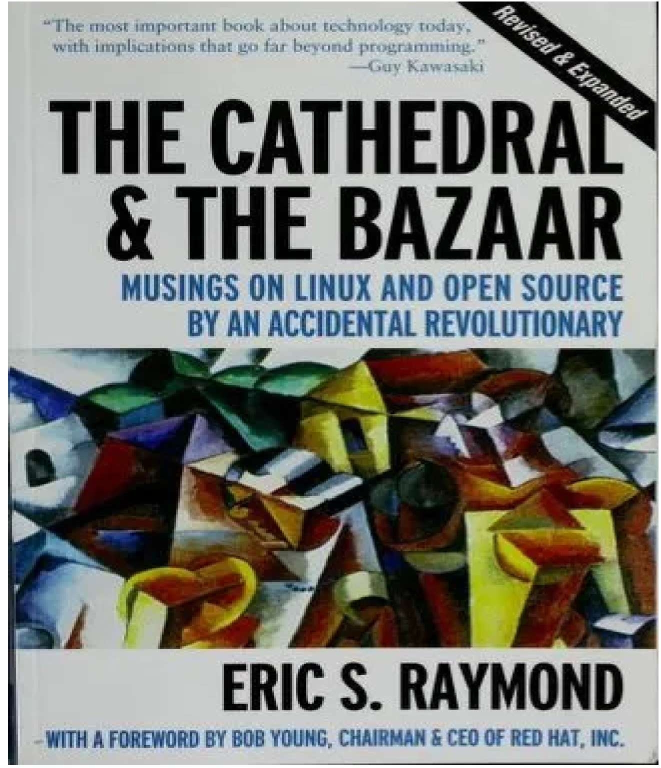
Ebook Publisher: http://freetamilebooks.com
Cover Image: Lenin Gurusamy - guruleninn@gmail.com
Ebook Creation: Iswarya Lenin - aishushanmugam09@gmail.com
Ebook Project: Kaniyam Foundation - kaniyam.com/foundation

பேராலயமும் சந்தையும்

மூலநூல்: The Cathedral and the Bazaar by Eric S. Raymond – Musings on Linux and Open Source by an Accidental Revolutionary – version 3.0

பேராலயமும் சந்தையும் – எரிக் ரேமண்ட் – ஒரு தற்செயலான புரட்சியாளரின் லினக்ஸ் மற்றும் திறந்த மூலம் பற்றிய சிந்தனைகள் – பதிப்பு 3.0

தமிழாக்கம்: இரா. அசோகன் ashokramach@gmail.com



பேராலயமும் சந்தையும் நூல்

முகவுரை: இதில் நீங்கள் ஏன் அக்கறை காட்ட வேண்டும்

உங்கள் கைகளில் உள்ள புத்தகம் கணினி நிரல் எழுதுவதில் வல்லுநர்கள் செயல்படும் முறை மற்றும் கலாச்சாரம் பற்றியது. இது நிரலாளர்கள் மற்றும் தொழில்நுட்ப மேலாளர்களுக்காக எழுதிய கட்டுரைகளின் தொகுப்பு. வாசகராகிய நீங்கள் கேட்கக்கூடிய கண்கூடான (எனினும் முற்றிலும் நியாயமான) கேள்வி: “நான் ஏன் இதில் அக்கறை காட்ட வேண்டும்?”

உலகப் பொருளாதாரத்தில் கணினி மென்பொருள் மிக முக்கியமான அம்சமாகி விட்டது

இந்தக் கேள்விக்குத் தெளிவான பதில் என்னவென்றால், உலகப் பொருளாதாரத்திலும் வணிக நிறுவனங்களின் வியூகக் கணக்கீடுகளிலும் கணினி மென்பொருள் மிக முக்கியமான அம்சமாகி விட்டது. இந்நூலை நீங்கள் படிக்கத் தொடங்கியிருக்கிறீர்கள் என்றால் தகவல் பொருளாதாரம், எண்ணிம யுகம் மற்றும் இணைய உலகம் பற்றிய இன்றைய பல உண்மைகளை நீங்கள் நிச்சயமாக அறிந்திருப்பீர்கள். நான் அவற்றை இங்கு திரும்பக் கூற மாட்டேன். சிறந்த தரமான, அதி நம்பகமான மென்பொருளை எவ்வாறு

உருவாக்குவது என்பது பற்றிய நமது புரிதலில் குறிப்பிடத்தக்க முன்னேற்றம் அடைந்தால் அது மிகப்பெரிய தாக்கங்களை ஏற்படுத்தும் என்பதை மட்டும் நான் சுட்டிக்காட்ட விரும்புகிறேன்.

திறந்த மூலம் செலவையும் குறைத்து மென்பொருள் தரத்தையும் மேம்படுத்துகிறது

இந்நூலில் உள்ள கட்டுரைகள் அத்தகைய அடிப்படை முன்னேற்றத்தைப் புதிதாகக் கண்டுபிடிக்கவில்லை, ஆனால் இவை ஒன்றை விவரிக்கின்றன. அதுதான் திறந்த மூல மென்பொருள். அதாவது திறந்த மேம்பாட்டை முறையாகப் பயன்படுத்துவதற்கான செயல்முறை மற்றும் செலவையும் குறைத்து மென்பொருள் தரத்தையும் மேம்படுத்துவதற்காக பரவலாக்கப்பட்ட சக ஆய்வு (decentralized peer review). திறந்த மூல மென்பொருள் ஒரு புதிதாக உதித்த யோசனை அல்ல (அதன் மரபுகள் முப்பது ஆண்டுகளுக்கு முன்பு இணையத்தின் தொடக்கத்திற்குச் செல்கின்றன). ஆனால் சமீபத்தில்தான் தொழில்நுட்ப மற்றும் சந்தை சக்திகள் ஒன்றிணைந்து அதை ஒரு குறுகிய பங்கிலிருந்து பரவலாக வெளியே கொண்டுவந்துள்ளன. இன்று திறந்த மூல இயக்கம் அடுத்த நூற்றாண்டின் கணினிசார் உள்கட்டமைப்பை (computing infrastructure) வரையறுக்க வலுவாக முனைகிறது. இது கணினியை நம்பியிருக்கும் யாவரும் புரிந்து கொள்ள வேண்டிய முக்கியமான சங்கதி.

கொந்தர்கள் **(hackers)** லினக்ஸ் மற்றும் பல திறந்த மூல மென்பொருட்களை உருவாக்குகிறார்கள்

நான் “திறந்த மூல இயக்கம்” என்று குறிப்பிட்டேன். வாசகர் இதில் அக்கறை காட்ட இன்னும் ஆர்வமூட்டும் காரணங்கள் உள்ளன என்பதை இது கோடி காட்டுகிறது. திறந்த மூலம்

என்ற நூதன எண்ணம் இந்த முப்பது வருடங்களாக இணையத்தைச் சார்ந்த ஒரு தீவிரமான மரபினரால் பின்பற்றப்பட்டு, நடைமுறையில் நிகழ்த்தப்பட்டு, போற்றப்பட்டது. தங்களைப் பெருமையுடன் “கொந்தர்கள் (hackers)” என்று அழைத்துக்கொள்ளும் நபர்கள் இவர்கள்தான். இப்போது ஊடகங்கள் இச்சொல்லைக் கணினிக் குற்றவாளி (computer criminal) என்று பொருள்பட முறை தவறிப் பயன்படுத்துகிறார்கள். இவர் ஓர் ஆர்வலர், ஒரு கலைஞர், ஒரு நோண்டி (tinkerer), பிரச்சினைகளுக்குத் தீர்வு காண்பவர், ஒரு நிபுணர்.

கொந்தர்களின் மரபினர், கடினமான தொழில்நுட்ப சிக்கல்கள் மற்றும் பெரும்பான்மையின் அலட்சியம் மற்றும் நிராகரிப்பு ஆகியவற்றிற்கு எதிராகப் பல பத்தாண்டுகளாகத் தெளிவற்ற நிலையில் போராடி, சமீபத்தில்தான் ஏற்றுக் கொள்ளப்பட்டனர். இவர்கள் இணையத்தை உருவாக்கினர்; இவர்கள் யூனிக்ஸ் (Unix) கட்டுமானம் செய்தனர்; இவர்கள் உலகளாவிய வலையை உருவாக்கினர்; இவர்கள் இன்று லினக்ஸ் மற்றும் திறந்த மூல மென்பொருளை உருவாக்குகிறார்கள். மேலும், 1990 களில் ஏற்பட்ட அதீத இணைய வளர்ச்சியைத் தொடர்ந்து, உலகின் பிற பகுதிகள் இறுதியாக இவர்கள் மீது முன்னதாகவே அதிக கவனம் செலுத்தியிருக்க வேண்டும் என்று கண்டறிந்துள்ளனர்.

எதிர்காலத்தில் வாழ்ந்து வேலை செய்ய வேண்டிய யாவருக்கும் கொந்தர் கலாச்சாரம் ஆர்வமூட்டும்

கொந்தர் கலாச்சாரம் மற்றும் அதன் வெற்றிகள் மனித உந்துதல், வேலையின் அமைப்பு, தொழில்முறையின் எதிர்காலம் மற்றும் நிறுவனத்தின் வடிவம் பற்றிய சில அடிப்படைக்

கேள்விகளை முன்வைக்கிறது. மேலும் 21 ஆம் நூற்றாண்டு மற்றும் அதற்கு அப்பால் உள்ள தகவல் வளம் மிகுந்த, பற்றாக்குறைக்குப் பிந்தைய பொருளாதாரங்களில் இவை அனைத்தும் எவ்வாறு மாறும் மற்றும் பரிணமிக்கும் என்ற கேள்வி எழுகிறது. மனிதகுலம் தங்கள் பொருளாதாரச் சூழலுடன் தொடர்பு கொள்ளும் மற்றும் மறுவடிவமைக்கும் விதத்தில் சில ஆழமான மாற்றங்களுக்கு முன்னோடியாகக் கொந்தர் கலாச்சாரம் இருக்கக்கூடும். கொந்தர் கலாச்சாரத்தைப் பற்றி நமக்குத் தெரிய வந்தது எதிர்காலத்தில் வாழ்ந்து வேலை செய்ய வேண்டிய யாவருக்கும் ஆர்வமூட்டுவதாக இருக்க வேண்டும்.

ஏதாவது உங்களுக்குப் புரியாவிட்டால் அவற்றைத் தயங்காமல் ஒதுக்கி முன்செல்லவும்

இந்நூல் முதலில் இணையத்தில் வெளியான கட்டுரைகளின் தொகுப்பு. கொந்தராட்சியின் சுருக்கமான வரலாறு (A Brief History of Hackerdom) முதலில் 1992 இல் வெளியிடப்பட்டது, ஆனால் தொடர்ந்து புதுப்பிக்கப்பட்டு திருத்தப்பட்டது. மற்றவை பிப்ரவரி 1997 மற்றும் மே 1999 க்கு இடையில் எழுதப்பட்டன. அவை அக்டோபர் 1999 இல் முதல் பதிப்பிற்காக ஓரளவு திருத்தப்பட்டு விரிவாக்கப்பட்டன. மேலும் ஜனவரி 2001 இரண்டாவது பதிப்பிற்காக மீண்டும் புதுப்பிக்கப்பட்டன. ஆனால் தொழில்நுட்ப சொல்லாட்சியை அகற்றவோ அல்லது பொது வாசகர்களுக்காக அவற்றை அதிகம் அணுகக்கூடியதாக (அறிவார்ந்த கருத்துகளை எளிதாக்குதல் போன்ற) தீவிர முயற்சி எதுவும் எடுக்கப்படவில்லை. வாசகர்களை சலிக்கவைத்து அவமானப்படுத்துவதை விட புதிர் போட்டு சவால் விடுவது மரியாதைக்குரியது என்று நான் நினைக்கிறேன். குறிப்பிட்ட தொழில்நுட்ப அல்லது

வரலாற்றுக் கருத்துகள் அல்லது ஏதாவதொரு கணினி அஃகுப்பெயர் (acronym) உங்களுக்குப் புரியாவிட்டால், அவற்றைத் தயங்காமல் ஒதுக்கி முன்செல்லவும். முழு நூலும் ஒரு கதையைச் சொல்கிறது. மேலும் உங்களுக்குப் பின்னர் தெரியவருவது முன்னர் உங்களுக்குப் புரியாததைத் தெளிவுபடுத்தக்கூடும்.

இக்கட்டுரைகள் உருவாகி வரும் ஆவணங்கள் என்பதையும் வாசகர் புரிந்து கொள்ள வேண்டும். இவற்றைப்பற்றிக் கருத்து எழுதும் அல்லது சரிசெய்யும் நபர்களின் பின்னூட்டங்களின் முக்கிய முடிவுகளை நான் அவ்வப்போது இணைக்கிறேன். இந்நூலில் ஏதேனும் பிழைகள் இருந்தால் அதற்கு நான் மட்டுமே பொறுப்பு. எனினும், இது திறந்த மூல மென்பொருள் போன்ற ஒரு சக மதிப்பாய்வு செயல்முறையால் பயனடைந்துள்ளது. மேலும் இங்கு பட்டியலிட முடியாத அளவுக்கு அதிகமான நபர்களின் பங்களிப்புகளை உள்ளடக்கியது. இப்பதிப்புகள் நிலையான அல்லது இறுதி வடிவங்கள் அல்ல. மாறாக, இவை விவரிக்கும் கலாச்சாரத்தின் பல உறுப்பினர்கள் பங்கேற்கும் தொடர்ச்சியான விசாரணையின் அறிக்கைகளாக இவை கருதப்பட வேண்டும்.

இறுதியாக, இந்நூலுக்கு வழிவகுத்த பலருக்கும் மற்றும் தொடர்ந்து நல்வாய்ப்பாகக் கூடிவந்த சூழ்நிலைகளுக்கும் எனது மகிழ்ச்சியையும் ஆச்சரியத்தையும் நன்றியையும் தெரிவித்துக் கொள்கிறேன்....

நீண்ட கால நட்பு மற்றும் இந்நூலின் உள்ளடக்க வேலைகளுக்கு ஆதரவளித்தவர்களுக்கு சில குறிப்பிட்ட நன்றிகள். நன்றி, லினஸ் டோர்வால்ட்ஸ் (Linus Torvalds). நன்றி, லேரி அகஸ்டின் (Larry Augustin). நன்றி, டாக் சியர்ல்ஸ் (Doc Searls). நன்றி, டிம் ஓ'ரெய்லி

(Tim O'Reilly). நீங்கள் அனைவரும் நண்பர்கள் மற்றும் உடன் பணியாற்றுபவர்கள் என்று அழைப்பதில் நான் பெருமைப்படுகிறேன். குறிப்பாக: நன்றி, கேத்தரின் ரேமண்ட் (Catherine Raymond) — என் அன்பு மனைவி மற்றும் எனது நீண்ட கால ஆதரவாளர்.

நான் ஒரு கொந்தர்

நான் ஒரு கொந்தர். இந்நூலில் விவரிக்கப்பட்டுள்ள கலாச்சாரத்தின் ஒரு பகுதியாக நான் இருபது ஆண்டுகளுக்கும் மேலாக இருக்கிறேன். அந்த நேரத்தில், உலகிலுள்ள சில சுவாரசியமான மற்றும் தனிச்சிறப்பு வாய்ந்த மனிதர்களுடன் வேலை செய்வதற்கும் விளையாடுவதற்கும், ஆர்வத்தைத் தூண்டும் சிக்கல்களைத் தீர்ப்பதற்கும் (ஒரு சில சந்தர்ப்பங்களில்) உண்மையிலேயே புதிய மற்றும் பயனுள்ளவற்றை உருவாக்குவதற்கும் எனக்குப் பாக்கியம் கிடைத்தது. மதிப்புமிக்க பாடங்கள், நாங்கள் பகிர்ந்த கைத்திறன் மற்றும் பல சங்கதிகளைப் பற்றி இங்கே பெயரிட இயலாத பலர் எனக்குக் கற்றுக் கொடுத்துள்ளனர். இந்நூலில் உள்ள கட்டுரைகள் அவர்களுக்கு எனது நன்றிக்கடன்.

இக்கட்டுரைகள் எனக்கும் கண்டுபிடிப்பின் கட்டங்களாக இருந்தன. இவை ஒரு கவர்ச்சிகரமான பயணத்தின் அறிக்கைகள் போன்றவை. இப்பயணத்தில் நீண்ட காலமாகப் பழகியவற்றைப் புதிய ஆழமான கண்ணோட்டத்தில் பார்க்கக் கற்றுக்கொண்டேன். இப்பயணத்தைப் பற்றிய என் அறிக்கைகள் திறந்த மூலம் பெரும்பான்மை நடைமுறையில் வருவதற்கு ஊக்கமளிக்கும் விளைவை ஏற்படுத்தியது என்பது அப்போது மட்டுமல்ல தொடர்ந்து எனக்கு பிரமிப்பை அளிக்கிறது. எனது பயண அனுபவங்களைப்

படிப்பவர்கள் இப்பயணத்தின் உற்சாகத்தையும் பெரும்பான்மை வணிக உலகும் நுகர்வோரும் இதே பாதையில் தங்கள் முதல் அடிகளை எடுத்து வைப்பதால் இன்று நம் முன் விரியும் அற்புதமான வாய்ப்புகளைப் பற்றிய உற்சாகத்தையும் பகிர முடியுமென்று நம்புகிறேன்.

நூல் சுருக்கம் 
லினக்ஸ் படைப்பாளர் மற்றும் தலைமை நிரலாளர் லினஸ் டோர்வால்ட்ஸ்

பேராலயமும் சந்தையும்

மென்பொருள் பொறியியல் பற்றிய சில ஆச்சரியமான கோட்பாடுகளை லினக்ஸின் (Linux) வரலாறு அறிவுறுத்தியது. இவற்றை சோதனை செய்வதற்காகவே நான் நடத்திய வெற்றிகரமான திறந்த மூல திட்டமான ஃபெட்ச்மெயிலைக் (fetchmail) கூறுபடுத்தி ஆய்வு செய்கிறேன். வணிக உலகில் பெரும்பாலும் “பேராலயம்” பாணியில்தான் மென்பொருள் உருவாக்கப்படுகிறது. லினக்ஸ் உலகின் “சந்தை” பாணி இதற்கு அடிப்படையிலேயே முற்றிலும் மாறுபட்டது. இந்த வேறுபட்ட வளர்ச்சிப் பாணிகளின் கோட்பாடுகளை நான் விவாதிக்கிறேன். இந்தப் பாணிகள் மென்பொருள் வழுநீக்கல் (debugging) பணியின் தன்மையைப் பற்றிய முற்றிலும் மாறுபட்ட அனுமானங்களிலிருந்து வந்தவை என்பதைக் காட்டுகிறேன். லினக்ஸ் அனுபவத்திலிருந்து நான் ஒரு நிலையான வாதத்தை முன்வைக்கிறேன், “அனேகம் பேர் கவனித்தால் அனைத்து வழுக்களும் எளியவையே (Given enough eyeballs, all bugs are shallow)”. தன்னல முகவர்களின் (selfish agents) தானாகவே திருத்திக் கொள்ளும் அமைப்புகளுடன் (self-correcting systems) ஆக்கவளமுடைய ஒப்புமைகளைப் பரிந்துரைக்கிறேன். மேலும் மென்பொருளின் எதிர்காலத்திற்கான இந்த நுண்ணறிவின் தாக்கங்கள் பற்றிய சில புத்தாய்வுகளுடன் முடிக்கிறேன்.

லினக்ஸ் புரட்சிகரமானது. ஐந்தாண்டுகளுக்கு முன்பு (1991) இணையம் வழியாக மட்டுமே தொடர்பிலிருக்கும் பல ஆயிரம் நிரலாளர்களின் பகுதி நேர வேலையில் ஓர் உலகத் தரம் வாய்ந்த இயங்குதளம் மந்திரம் செய்தது போல ஒன்றுகூடி வரும் என்று யார்தான் நினைத்திருக்க முடியும்?

நிச்சயமாக நான் அல்ல. 1993 ஆம் ஆண்டின் தொடக்கத்தில் லினக்ஸ் எனக்குத் தெரிய வந்தபோது, நான் ஏற்கனவே பத்தாண்டுகளாக யூனிக்ஸ் (Unix) மற்றும் திறந்த மூல மேம்பாட்டில் ஈடுபட்டிருந்தேன். 1980களில் குனு (GNU) வின் முதல் பங்களிப்பாளர்களில் நானும் ஒருவன். இன்றும் பரவலாகப் பயன்பாட்டில் உள்ள பல திறந்த மூல நிரல்களை (nethack, Emacs VC மற்றும் GUD mods, xlife இன்ன பிற) உருவாக்கி அல்லது இணைந்து உருவாக்கி வெளியிட்டிருந்தேன். அதை எப்படிச் செய்வதென்று எனக்குத் தெரியுமென்று நினைத்தேன்.

எனக்குத் தெரியும் என்று நான் நினைத்த பலவற்றை லினக்ஸ் தலைகீழாகப் புரட்டிப் போட்டது. நான் பல ஆண்டுகளாக சிறிய கருவிகள், விரைவான முன்மாதிரி மற்றும் பரிணாம நிரலாக்கம் (evolutionary programming) போன்ற யூனிக்ஸ் நற்செய்தியைப் பரப்பிக் கொண்டிருந்தேன்.

சிக்கலான மென்பொருட்களுக்கு பேராலயங்களைப் போல மையக் கட்டுப்பாடு அவசியம் என்று நான் நம்பினேன்

ஆனால் ஒரு குறிப்பிட்ட அளவுக்கு மேல் மிகச்சிக்கலான திட்டங்களுக்கு மையக் கட்டுப்பாட்டிலுள்ள திட்டமிட்ட அணுகுமுறை தேவை என்று நம்பினேன். மிக முக்கியமான

மென்பொருட்கள் (இயங்கு தளங்கள் மற்றும் ஈமாக்ஸ் (Emacs) நிரல் தொகுப்பி போன்ற பெரிய கருவிகள்) பேராலயங்களைப் போல உருவாக்கப்பட வேண்டும் என்று நான் நம்பினேன். தேர்ச்சித்திறன் கொண்ட தனி நிபுணர்கள் அல்லது சிறிய குழுக்கள் மூலம் கவனமாக வடிவமைக்கப்பட வேண்டும் என்று நம்பினேன். முழுமை அடையாத பீட்டா (beta) நிலையில் மென்பொருளை வெளியிடக் கூடாது என்றும் நினைத்தேன்.

இரைச்சல் மிகுந்த சந்தைக்கடையாக இருந்த லினக்ஸ் சமூகம் மிக நன்றாக வேலை செய்தது ஆச்சரியமாக இருந்தது

லினக்ஸ் உருவாக்குனர் மற்றும் தலைமை நிரலாளர் லினஸ் டோர்வால்ட்ஸின் (Linus Torvalds) வளர்ச்சியின் பாணி— முன்னதாக வெளியிடுதல் அடிக்கடி வெளியிடுதல், முடிந்த அளவு அனைத்துப் பொறுப்புகளையும் பகிர்ந்தளித்தல், முற்றிலுமாகத் திறந்திருத்தல்—ஆச்சரியமாக இருந்தது. இங்கு எவரும் மிக அமைதியாக, பயபக்தியுடன் பேராலயம் கட்டவில்லை. மாறாக, லினக்ஸ் சமூகம் மாறுபட்ட குறிக்கோள்கள் மற்றும் அணுகுமுறைகள் கொண்ட இரைச்சல் மிகுந்த சந்தைக்கடையைப் போல் இருந்தது. எடுத்துக்காட்டாக, லினக்ஸ் காப்பக தளங்களில் எவர் வேண்டுமானாலும் நிரல் சமர்ப்பிக்கலாம். இதிலிருந்து ஓர் ஒத்திசைவான மற்றும் நிலையான மென்பொருள் வெளிவர வேண்டுமென்றால் வரிசையாகப் பல அற்புதங்கள் நிகழ வேண்டும் போலிருந்தது.

இந்த சந்தை பாணி வேலை செய்கிறது, அதுவும் நன்றாக வேலை செய்கிறது என்பது ஓர் அளப்பரிய அதிர்ச்சியை ஏற்படுத்தியது. நான் அதில் நுழைந்த போது, அதன் தனிப்பட்ட திட்டங்களில் கடுமையாக வேலை செய்தேன். மேலும் லினக்ஸ் உலகம் ஏன்

குழப்பத்தில் பிய்த்துக் கொள்ளவில்லை என்பது மட்டுமல்லாமல் பேராலயம் கட்டுபவர்கள் கற்பனை கூட செய்ய முடியாத வேகத்தில் வலிமையிலிருந்து வலிமைக்குச் சென்றது ஏன் என்பதைப் புரிந்துகொள்ள இன்னும் கடுமையாக முயற்சித்தேன்.

சந்தை பாணியில் ஒரு திறந்த மூல திட்டத்தை இயக்கும் நல்ல வாய்ப்பு எனக்குக் கிடைத்தது

1996 நடுவில் நான் ஓரளவு புரிந்து கொள்ளத் தொடங்கினேன். எனது கோட்பாட்டைச் சோதிக்க ஒரு நல்ல வாய்ப்பு எனக்குக் கிடைத்தது. அது சந்தை பாணியில் நான் உணர்வுபூர்வமாக இயக்கிப் பார்க்கக்கூடிய ஒரு திறந்த மூல திட்ட வடிவில் வந்தது. அத்திட்டத்தில் என் கோட்பாட்டைச் செயல்படுத்தினேன். அது ஒரு குறிப்பிடத்தக்க வெற்றியானது.

சந்தை பாணியில் நான் இயக்கிய அந்தத் திறந்த மூல திட்டத்தின் கதைதான் இந்நூல்

இதுதான் அத்திட்டத்தின் கதை. திறம்பட்ட திறந்த மூல மென்பொருள் உருவாக்கம் பற்றிய சில >மணிமொழிகளை முன்மொழிய இதைப் பயன்படுத்துகிறேன். இவை அனைத்தும் லினக்ஸ் உலகில் நான் முதன்முதலில் கற்றுக்கொண்ட சங்கதிகள் அல்ல. ஆனால் லினக்ஸ் உலகம் இவற்றுக்கு எவ்வாறு குறிப்பிட்ட முக்கியத்துவத்தை அளிக்கிறது என்பதைப் பார்ப்போம். தொடர்ந்து நல்ல மென்பொருள் வழங்கும் ஊற்றாக லினக்ஸ் சமூகத்தை ஆக்குவது எது என்பதைச் சரியாகப் புரிந்துகொள்ள இவை உங்களுக்கு உதவும் என்று நம்புகிறேன். ஒருவேளை இவை உங்கள் உற்பத்தித் திறனையும் அதிகரிக்க உதவலாம்.



பேராலயமும் சந்தையும் மூல நூலாசிரியர் எரிக் ரேமண்ட்

அஞ்சல் போய்ச் சேர்ந்தாக வேண்டும்

1993 ஆம் ஆண்டு முதல் பென்சில்வேனியாவின் வெஸ்ட் செஸ்டரில் செஸ்டர் கவுண்டி இண்டர்லிங்க் (CCIL) என்ற சிறிய இலவச இணைய சேவையகத்தின் தொழில்நுட்ப வேலையைச் செய்து வருகிறேன். நான் CCIL இன் நிறுவனர்களில் ஒருவன். எங்களின் தனித்துவமான பல பயனர் அறிக்கைப் பலகை (multiuser bulletin**oard) மென்பொருளையும் நானே எழுதினேன். நீங்கள் அதை locke.ccil.org க்கு டெல்நெட் (telnet) செய்து பார்க்கலாம். இன்று இது முப்பது இணைப்புகளில் சுமார் மூவாயிரம் பயனர்களை ஆதரிக்கிறது. இந்த வேலை CCIL இன் 56K தொடர்பு மூலம் 24 மணி நேரமும் இணையத்தை அணுக என்னை அனுமதித்தது. உண்மையில், இந்த வேலை அதைக் கோரியது!

நான் உடனடி இணைய மின்னஞ்சலுக்கு மிகவும் பழகிவிட்டேன். எனக்கு மின்னஞ்சல் ஏதாவது வந்ததா என்று பார்க்க அவ்வப்போது டெல்நெட் மூலம் லாக் இல் (locke.ccil.org) புகுபதிகை செய்வது எரிச்சலூட்டுவதாக இருந்தது. நான் விரும்பியது என்னவென்றால் எனது அஞ்சல் எனது வீட்டு அமைப்பான ஸ்னார்க்கில் (snark) வந்து சேர்ந்துவிடவேண்டும். வந்தவுடன் எனக்கு அறிவிப்பு வரும். எனது கணினியில் உள்ள எல்லாக் கருவிகளையும் பயன்படுத்தி என்னால் அதைக்

கையாள முடியும்.

இணையத்தின் வழக்கமான அஞ்சலை மேலனுப்பும் நெறிமுறை SMTP (Simple Mail Transfer Protocol) இந்த வேலைக்குத் தோதாகாது. ஏனெனில் இதற்குக் கணினிகள் முழுநேரமும் இணையத்தில் இருக்க வேண்டும். எனது கணினி எப்போதும் இணையத்தில் இருக்காது. மேலும் அதற்கு நிலையான ஐபி (IP) முகவரியும் கிடையாது. எனக்குத் தேவையானது என்னவென்றால் எனது இடைவிட்ட அழைப்புவழி இணைப்பு (intermittent dialup connection) மூலம் தொடர்புகொண்டு எனது மின்னஞ்சலை இழுத்துக் கொண்டு வந்து சேர்க்கும் ஒரு நிரலாகும். இதுபோன்ற நிரல்கள் இருப்பதும் மேலும் அவற்றில் பெரும்பாலானவை POP (Post Office Protocol) என்ற எளிய பயன்பாட்டு நெறிமுறையைப் பயன்படுத்துகின்றன என்பதும் எனக்குத் தெரியும். POP இப்போது பொதுவான அஞ்சல் பயனர் மென்பொருட்களால் பரவலாக ஆதரிக்கப்படுகிறது. ஆனால் அந்தக் காலகட்டத்தில் என்னிடமிருந்த அஞ்சல் மென்பொருளில் அந்த அம்சம் இல்லை.

எனக்கு POP3 பயனர் மென்பொருள் தேவைப்பட்டது. எனவே நான் இணையத்தில் தேடி ஒன்றைக் கண்டுபிடித்தேன். உண்மையில் நான் மூன்று அல்லது நான்கைக் கண்டுபிடித்தேன். அவற்றில் ஒன்றைச் சில காலம் பயன்படுத்தினேன். ஆனால் அதில் ஓர் அவசியமான அம்சம் இல்லை. பதிலஞ்சல் சரியான முகவரிக்குத் திரும்பிப் போய்ச்சேருமாறு வந்த மின்னஞ்சலில் உள்ள முகவரிகளைக் கொந்தும் (hack) திறன் இல்லை.

பிரச்சினை இதுதான். லாக் (locke.ccil.org) இல் ‘ஜோ’ என்ற ஒருவர் எனக்கு மெயில் அனுப்பினார் என்று வைத்துக்கொள்வோம். நான் ஸ்நார்க்கில் மின்னஞ்சலைப் பெற்று, அதற்குப் பதிலளிக்க

முயன்றால், எனது அஞ்சல் மென்பொருள் அதை ஸ்னார்க்கில் இல்லாத ‘ஜோ’ வுக்கு அனுப்புவதற்கு முயற்சி செய்யும். இதைக் கையாள்வதற்காக பதில் முகவரி ஒவ்வொன்றிலும் <@ccil.org> சேர்த்துக் கையால் திருத்துவது பெரிய தலைவலியாக ஆகியது.

இது நிச்சயமாக கணினி எனக்குச் செய்து தரவேண்டிய வேலை. ஆனால் அப்போதிருந்த POP பயனர் மென்பொருள் எதற்கும் இதை எப்படிச் செய்வதென்று தெரியவில்லை! இதுதான் நமக்கு முதல் பாடம்:

> மணிமொழி 1. எந்தவொரு நல்ல மென்பொருள் திட்டமும் நிரலாளரின் சொந்தத் தலைவலிகளைத் தீர்த்து வைப்பதன் மூலம்தான் தொடங்குகிறது.

தேவையே கண்டுபிடிப்பின் தாய்

இது கண்கூடாகத் தெரிந்திருக்க வேண்டும். “தேவையே கண்டுபிடிப்பின் தாய் (Necessity is the mother of invention)” என்பது நீண்ட காலமாக உள்ள பழமொழிதானே. ஆனால் பெரும்பாலும் மென்பொருள் உருவாக்குநர்கள் தங்களுக்குத் தேவையில்லாத மற்றும் விரும்பாத திட்டங்களில் ஊதியத்திற்காகத் தங்கள் நாட்களைக் கழிக்கிறார்கள். ஆனால் லினக்ஸ் உலகில் இவ்வாறு இல்லை. இது லினக்ஸ் சமூகத்தில் உருவாகும் மென்பொருட்களின் சராசரி தரம் ஏன் உயர்வாக உள்ளது என்பதை விளக்கக்கூடும்.

எனவே, ஏற்கனவே உள்ளவற்றுடன் போட்டியிடும் வகையில் நான் உடனடியாகப் புத்தம் புதிய POP3 பயனர் மென்பொருளை ஆவேசமாக எழுதத் தொடங்கினேனா? நிச்சயமாக இல்லை! நான்

கையில் வைத்திருந்த POP பயன்பாட்டு நிரல் தொடர்களைக் கவனமாகப் பார்த்தேன். “நான் விரும்புவதற்கு மிக நெருக்கமானது எது?” என்பதுதான் கேள்வி. ஏனென்றால்:

> மணிமொழி 2. நல்ல நிரலாளர்களுக்கு என்ன எழுதுவது என்று தெரியும். தலை சிறந்த நிரலாளர்களுக்கு எதை எடுத்துத் தனக்குத் தோதாக மாற்றி எழுதிக்கொள்வது என்று (மீண்டும் பயன்படுத்தவும்) தெரியும்.

தலை சிறந்த நிரலாளர்களின் முக்கியமான பண்பு ஆக்கப்பூர்வமான சோம்பல்

நான் ஒரு தலை சிறந்த நிரலாளர் என்று கூறிக்கொள்ளவில்லை. எனினும் ஒருவரைப் போல் பாவிக்க முயற்சிக்கிறேன். தலை சிறந்த நிரலாளர்களின் முக்கியமான பண்பு ஆக்கப்பூர்வமான சோம்பல். நல்ல பெயர் பெறுவது முயற்சிக்காக அல்ல, முடிவுகளுக்காகத்தான் என்பது அவர்களுக்குத் தெரியும். முதலில் இருந்து தொடங்குவதைக் காட்டிலும் ஒரு நல்ல பகுதியளவு தீர்விலிருந்து தொடங்குவது எப்போதுமே எளிதானது என்பதையும் அவர்கள் அறிவார்கள்.

எடுத்துக்காட்டாக, லினக்ஸை (Linux) முதலில் இருந்து எழுத லினஸ் டோர்வால்ட்ஸ் (Linus Torvalds) முயற்சிக்கவில்லை. மாறாக, தனிநபர் கணினிகளுக்கான யூனிக்ஸ் போன்ற ஆனால் சிறிய இயங்குதளமான மினிக்ஸின் (Minix) நிரல் மற்றும் யோசனைகளை மீண்டும் பயன்படுத்துவதன் மூலம்தான் தொடங்கினார். இறுதியில் அனைத்து மினிக்ஸ் நிரல்களும் வழக்கொழிந்து போயின அல்லது முழுமையாக மீண்டும் எழுதப்பட்டன. எனினும் அது இருந்தவரை இறுதியில் லினக்ஸாக ஆகிய படரத் தொடங்கிய கொடியை மினிக்ஸ்

பந்தலாகத் தாங்கியது.

அதே நோக்கத்தில், அடித்தளமாகப் பயன்படுத்த, ஓரளவு நல்ல நிரல் கொண்ட ஏற்கனவே உள்ள POP செயலியைத் தேடினேன்.

யூனிக்ஸ் / லினக்ஸ் உலகின் மூலப் பகிர்வு பாரம்பரியம் எப்போதும் நிரல் மறுபயன்பாட்டிற்கு ஒத்தாசையாகவே இருந்து வருகிறது

யூனிக்ஸ் உலகின் மூலப் பகிர்வு பாரம்பரியம் எப்போதும் நிரல் மறுபயன்பாட்டிற்கு ஒத்தாசையாகவே இருந்து வருகிறது. இதனால்தான் குனு (GNU) திட்டம் யூனிக்ஸை அடிப்படை OS ஆகத் தேர்ந்தெடுத்தது, அதைப் பற்றிய தீவிர ஐயப்பாடுகள் இருந்தபோதிலும். லினக்ஸ் உலகம் இப்பாரம்பரியத்தை ஏறக்குறைய அதன் தொழில்நுட்ப எல்லைக்குக் கொண்டு சென்றுள்ளது. டெராபைட்டுகள் (terabytes) அளவில் திறந்த மூல நிரல்கள் பொதுவாகக் கிடைக்கின்றன. எனவே லினக்ஸ் உலகில் மற்றவர்களின் ஓரளவு நல்ல நிரலைத் தேடுவதில் நேரத்தைச் செலவிடுவது உங்களுக்கு வேறு எங்கும் இல்லாத நல்ல பலனைத் தரும்.

எனக்கும் அது பலன் தந்தது. நான் முன்பு கண்டறிந்தவைகளையும் சேர்த்து, எனது இரண்டாவது தேடலில் fetchpop, PopTart, get-mail, gwpop, pimp, pop-perl, popc, popmail மற்றும் upop ஆக மொத்தம் ஒன்பது சாத்தியக்கூறுகள் கிடைத்தன. நான் முதன்முதலில் தேர்வு செய்தது சூங்-ஹோங் ஓ (Seung-Hong Oh) எழுதிய ஃபெட்ச்பாப் (fetchpop). நான் அதில் எனது தலைப்பு திரும்ப எழுதும் (header-rewrite) அம்சத்தைச் சேர்த்தேன், மேலும் பல மேம்பாடுகளையும் செய்தேன். அதை அதன் படைப்பாளர் தனது 1.9 வெளியீட்டில் ஏற்றுக்கொண்டார்.

சில வாரங்களுக்குப் பிறகு, கார்ல் ஹாரிஸின் பாப்கிளையன்ட் (popclient) நிரல் எதிர்பாராத விதமாக எனக்குத் தெரியவந்தது. இதனால் எனக்கு ஒரு பிரச்சினை எழுந்தது. ஃபெட்ச்பாப்பில் பின்னணியில் இயங்கும் சேவை நிரல் (background-daemon) போன்ற சில நல்ல புதுமையான யோசனைகள் இருந்தபோதிலும், அது POP3 ஐ மட்டுமே கையாள முடியும். மற்றும் அந்த நிரல் ஒரு கற்றுக்குட்டி எழுதியது போலிருந்தது (சூங்-ஹோங் அந்த நேரத்தில் ஒரு மதிநுட்பம் வாய்ந்த ஆனால் அனுபவமற்ற நிரலாளராக இருந்தார். இந்த இரண்டு பண்புகளும் நிரலில் தெரிந்தது). கார்லின் நிரல் சிறப்பாக இருந்தது, மிகவும் தொழில்முறைத் திறன் கொண்டது மற்றும் திடமானது. ஆனால் அவரது திட்டத்தில் பல முக்கியமான மற்றும் செயல்படுத்தச் சிக்கலான ஃபெட்ச்பாப் அம்சங்கள் இல்லை (நானே எழுதிய நிரல் உட்பட).

ஃபெட்ச்பாப்பிலேயே தொடர்வதா அல்லது மாறுவதா? ஒரு சிறந்த மேம்பாட்டு அடிப்படைத் தளத்திற்கு நான் மாறினால் அதன் விளைவாக நான் ஏற்கனவே செய்த நிரல்களைத் தூக்கி எறியவேண்டி வரும்.

மாறுவதற்கு ஒரு நடைமுறை நோக்கம் பல-நெறிமுறை ஆதரவு (multiple-protocol support) இருந்தது ஆகும். POP3 என்பது அஞ்சல்நிலைய வழங்கி நெறிமுறைகளில் பொதுவாகப் பயன்படுத்தப்படுகிறது, ஆனால் அது மட்டும்தான் என்று அல்ல. ஃபெட்ச்பாப் மற்றும் பிற போட்டி செயலிகள் POP2, RPOP அல்லது APOP ஐச் செய்யவில்லை. மேலும் வேடிக்கைக்காக IMAP (Internet Message Access Protocol, மிகச் சமீபத்தில் வடிவமைக்கப்பட்ட மற்றும் மிகவும் சக்திவாய்ந்த அஞ்சல்நிலைய நெறிமுறை) ஐச் சேர்ப்பது பற்றிய லேசான யோசனையும் எனக்கு ஏற்கனவே இருந்தது.

ஆனால் லினக்ஸுக்கு வெகு காலத்திற்கு முன்பே நான் கற்ற ஒன்று, மாறுவது ஒரு நல்ல யோசனையாக இருக்கலாம் என்று நினைக்க, எனக்குக் கோட்பாடு அடிப்படையில் ஒரு காரணமாக இருந்தது.

> மணிமொழி 3. முதல் முயற்சியைத் தூக்கி எறியத் திட்டமிடுங்கள்; எப்படியும் நீங்கள் அதைச் செய்யவேண்டி வரும். (ஃப்ரெட் புரூக்ஸ் (Fred Brooks), ஒரு நபரின் ஒரு மாத வேலை என்ற கட்டுக்கதை (The Mythical Man-Month), அத்தியாயம் 11)

முதல் முறை ஒரு தீர்வைச் செயல்படுத்தும் வரை பிரச்சினையை முழுமையாகப் புரிந்து கொள்ள மாட்டீர்கள்

இதையே வேறு விதமாகச் சொல்வதென்றால், முதல் முறை ஒரு தீர்வைச் செயல்படுத்தும் வரை பிரச்சினையை முழுமையாகப் புரிந்து கொள்ள மாட்டீர்கள். இரண்டாவது முறை, அதைச் சரியாகச் செய்யும் அளவுக்கு உங்களுக்குப் புரியக்கூடும். எனவே அதை ஒழுங்காகச் செய்ய விரும்பினால், குறைந்தபட்சம் ஒரு முறையாவது திரும்பத் தொடங்கத் தயாராக இருங்கள்.

சரி (என்று எனக்கே சொல்லிக் கொண்டேன்), ஃபெட்ச்பாப்பில் செய்த மாற்றங்கள் எனது முதல் முயற்சி. ஆகவே நான் பாப்கிளையன்டுக்கு மாறினேன்.

நான் 25 ஜூன் 1996 அன்று கார்ல் ஹாரிஸுக்கு எனது முதல் பாப்கிளையன்ட் ஒட்டு நிரல்களை (patches) அனுப்பிய பிறகுதான் தெரிய வந்தது, அவர் பாப்கிளையன்ட் மீதான ஆர்வத்தைச் சில காலத்திற்கு முன்பே இழந்துவிட்டார் என்பது. நிரல்

சிறிது சுத்தம் செய்யப்படாமல் இருந்தது, சிறிய வழுக்களில் கவனம் செலுத்தப்படவில்லை. நான் பல மாற்றங்களைச் செய்ய வேண்டியிருந்தது. அடுத்து நான் செய்ய வேண்டிய நியாயமான வேலை பாப்கிளையன்ட் நிரலின் பொறுப்பை முழுமையாக எடுத்துக்கொள்வது என்ற முடிவுக்கு நாங்கள் இருவரும் விரைவில் வந்தோம்.

நான் உண்மையில் கவனம் செலுத்தாமலேயே திட்டம் விரிவடைந்துவிட்டது. தற்போதுள்ள பாப்கிளையன்டிற்கான சிறிய ஒட்டு நிரல்களைப் பற்றிமட்டும் நான் இப்போது சிந்திக்கவில்லை. நான் அதை முழுவதுமாகப் பராமரிக்கும் பொறுப்பை ஏற்றுக்கொண்டேன். மேலும் பெரிய மாற்றங்களுக்கு வழிவகுக்கும் என்று எனக்குத் தோன்றிய பல யோசனைகள் என் தலையில் குமிழ்ந்தன.

உங்கள் மனப்பாங்கு சரியாக இருந்தால், சுவாரசியமான திட்டங்கள் உங்களைத் தானே தேடி வரும்

நிரல் பகிர்வை ஊக்குவிக்கும் மென்பொருள் கலாச்சாரத்தில், இது ஒரு திட்டம் பரிணமிக்கும் இயற்கையான வழியாகும். நான் பின்வரும் கொள்கைக்கு எடுத்துக்காட்டாக இருந்தேன்:

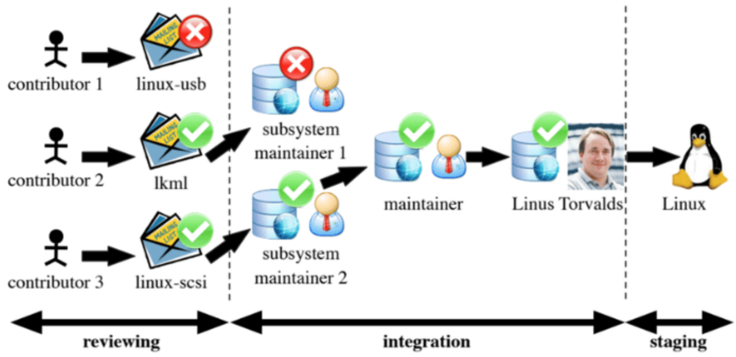
> மணிமொழி 4. உங்கள் மனப்பாங்கு சரியாக இருந்தால், சுவாரசியமான திட்டங்கள் உங்களைத் தானே தேடி வரும்.

ஆனால் கார்ல் ஹாரிஸின் மனப்பாங்கு அதைவிட முக்கியமானது. அவர் இந்த மணிமொழியைப் புரிந்து கொண்டார்

> மணிமொழி 5. நீங்கள் ஒரு திட்டத்தில் ஆர்வத்தை இழக்கும்போது, அதை ஒரு

தகுதிவாய்ந்த வாரிசிடம் ஒப்படைப்பதே அத்திட்டத்துக்கு நீங்கள் செய்யும் கடைசிக் கடமையாகும்.

அதைப் பற்றி நாங்கள் கலந்துரையாடவில்லை, எனினும் சிறந்த தீர்வை செயல்படுத்த வேண்டும் என்ற பொதுவான குறிக்கோள் இருவருக்கும் இருந்தது. நான் ஒரு நம்பத்தகுந்த நபர்தான் என்பதை நிறுவ முடியுமா என்பதுதான் எங்களுக்குள் ஒரே கேள்வி. நான் அதைச் செய்தவுடன், அவர் பெருந்தன்மையுடன் தாமதமின்றிப் பராமரிக்கும் பொறுப்பை என்னிடம் ஒப்படைத்தார். என் முறை வரும்போது நானும் இதேயளவு சிறப்பாக நடந்து கொள்வேன் என்று நம்புகிறேன்.



லினக்ஸ் கருநிரல் மேம்பாட்டுச் செயல்முறை

பயனர்கள் இருப்பதன் முக்கியத்துவம்

பயனர்களைச் சரியாகப் பண்படுத்தினால் அவர்கள் இணை உருவாக்குநர்களாகவும் ஆகலாம்

இப்படியாக நான் பாப்கிளையன்ட்டைப் பெற்றேன். அதைவிட முக்கியமாக, நான் பாப்கிளையன்ட்டின் பயனர் அடித்தளத்தைப் பெற்றேன். பயனர்கள் நமக்குத் தேவையான அற்புதமான நபர்கள். நாம் ஓர் உண்மையான தேவைக்குச் சேவை செய்கிறோம், எதையோ சரியாகச் செய்துள்ளோம் என்பதை அவர்கள் நிரூபிப்பதால் மட்டுமல்ல. சரியாகப் பண்படுத்தினால், அவர்கள் இணை உருவாக்குநர்களாகவும் ஆகலாம்.

யூனிக்ஸ் பாரம்பரியத்தின் மற்றொரு பலம், பல பயனர்கள் கொந்தர்களாகவும் (hackers) உள்ளனர். இதையே லினக்ஸ் நல்ல உச்சநிலைக்குக் கொண்டு சென்றுவிட்டது. மூல நிரல் கிடைப்பதால், அவர்கள் திறனான கொந்தர்களாக இருக்க இயலும். வழுத்திருத்த நேரத்தைக் குறைக்க இது மிகவும் பயனுள்ளதாக இருக்கும். கொஞ்சம் ஊக்கமளித்தால், உங்கள் பயனர்கள் சிக்கல்களைக் கண்டறிவார்கள், திருத்தங்களைப் பரிந்துரைப்பார்கள் மற்றும் நீங்கள் தனியாகப் போராடுவதைவிட விரைவாக நிரலை மேம்படுத்த உதவுவார்கள்.

> மணிமொழி 6. உங்கள் பயனர்களை இணை-நிரலாளர்களாகக் கருதுவது, விரைந்த நிரல் மேம்பாடு மற்றும் திறனான வழுத்திருத்தத்திற்கான பிரச்சினையற்ற வழியாகும்.

இந்த விளைவின் சக்தியை நாம் எளிதாகக் குறைத்து மதிப்பிடக்கூடும். உண்மையில், திறந்த மூல உலகில் நாம் அனைவரும், லினஸ் டோர்வால்ட்ஸ் (Linus Torvalds) செயல்படுத்திக் காண்பிக்கும் வரை, பயனர்களின் எண்ணிக்கை கூடி, திட்டமும் சிக்கலாகும்போது இது எவ்வளவு நன்றாகச் செயல்படுகிறது என்பதைக் கடுமையாகக் குறைத்து மதிப்பிட்டோம்.

தோல்வியடைய முடியாத அளவுக்குச் சோம்பேறி!

உண்மையில் லினஸின் சாமர்த்தியமான மற்றும் மிகவும் குறிப்பிடத்தக்கக் கொந்தல் (hack) லினக்ஸ் கருநிரலின் (kernel) கட்டுமானம் அல்ல. மாறாக அவர் லினக்ஸ் உருவாக்கும் பாணியைக் கண்டுபிடித்ததுதான் என்று நான் நினைக்கிறேன். ஒருமுறை அவர் முன்னிலையில் இக்கருத்தை நான் தெரிவித்தபோது சிரித்துக்கொண்டே, அவர் அடிக்கடி சொல்வதையே அமைதியாகச் சொன்னார்: “அடிப்படையில் நான் மிகவும் சோம்பேறி. மற்றவர்கள் செய்யும் வேலையை வைத்து நான் நல்ல பெயர் வாங்க விரும்புபவன்.” குள்ளநரியைப் போல சோம்பேறி. அல்லது, ராபர்ட் ஹெய்ன்லைன் (Robert Heinlein) தனது கதாபாத்திரங்களில் ஒன்றைப் பற்றிப் பிரபலமாக எழுதியது போல, தோல்வியடைய முடியாத அளவுக்குச் சோம்பேறி.

பின்னோக்கிப் பார்த்தால், லினக்ஸின் வழிமுறைகள் மற்றும் வெற்றிக்கான ஒரு முன்னுதாரணத்தை GNU Emacs Lisp நிரலகம் மற்றும் Lisp நிரல் காப்பகங்களின் வளர்ச்சியில் காணலாம்.

Emacs C core மற்றும் பிற குனு கருவிகளின் பேராலயம்-கட்டமைப்பு பாணிக்கு மாறாக, லிஸ்ப் நிரலகத்தின் பரிணாமம் நெகிழ்வானது மற்றும் பயனர்களால் வெகுவாக உந்தப்பட்டது. யோசனைகள் மற்றும் முன்மாதிரி முறைகள் நிலையான இறுதி வடிவத்தை அடைவதற்கு முன்பு மூன்று அல்லது நான்கு முறை திரும்ப எழுதப்பட்டன. மற்றும் லினக்ஸ் போலவே இணையம் மூலம் நெகிழ்வான ஒத்துழைப்புகள் அடிக்கடி இருந்தன.

உண்மையில், ஃபெட்ச்மெயிலுக்கு முந்தைய எனது சொந்த வெற்றிகரமான ஒற்றை முயற்சி அநேகமாக ஈமாக்ஸ் விசி (பதிப்புக் கட்டுப்பாடு) பயன்முறையாக (Emacs VC (version control) mode) இருக்கலாம். இதில் லினக்ஸ் போலவே மூன்று நபர்களுடன் மின்னஞ்சல் மூலம் கூட்டுப்பணியாற்றினேன். இவர்களில் ஒருவரை மட்டுமே (இமாக்ஸ் படைப்பாளரும் கட்டற்ற மென்பொருள் அறக்கட்டளை நிறுவனருமான ரிச்சர்ட் ஸ்டால்மேன்(Richard Stallman)), நான் இதுநாள்வரை சந்தித்துள்ளேன்.

இது SCCS, RCS மற்றும் பிற்கால CVS ஆகியவற்றிற்கான பயனர் முகப்பாக இருந்தது. இது எளிய பதிப்புக் கட்டுப்பாட்டு செயல்பாடுகளை வழங்கிய Emacs க்குள் இருந்து வந்தது. இது வேறு யாரோ எழுதிய ஒரு சிறிய, செப்பம் செய்யாத sccs.el பயன்முறையில் இருந்து உருவானது. மேலும் VC இன் வளர்ச்சி வெற்றியடைந்தது. ஏனெனில், Emacs போல் இல்லாமல், வெளியிடு/சோதனை செய்/மேம்படுத்து சுழற்சிகள் மூலம் மிக விரைவாக Emacs Lisp நிரலின் புதிய பதிப்புகளை வெளியீடு செய்ய முடியும்.

பேராலயம்-பாணி உட்கருவையும் சந்தை-பாணி கருவிப்பெட்டியையும் கொண்ட மென்பொருட்கள்

ஈமாக்ஸ் கதை தனித்துவமானது அல்ல.

பேராலயம்-பாணி உட்கருவையும் சந்தை-பாணி கருவிப்பெட்டியையும் இணைக்கும் இரண்டு-நிலைக் கட்டமைப்பு மற்றும் இரண்டு-அடுக்கு பயனர் சமூகம் கொண்ட பிற மென்பொருள் தயாரிப்புகள் உள்ளன. வணிகரீதியான தரவு பகுப்பாய்வு மற்றும் காட்சிப்படுத்தல் (data-analysis and visualization) கருவியான மேட்லாப் (MATLAB) என்பது அத்தகைய ஒன்றாகும். முக்கிய நடவடிக்கைகள், எழுச்சியும் மாற்றப்போக்கும், புதுப்புனைவு ஆகியவை பெரும்பாலும் கருவியின் திறந்த பகுதியில் நிகழ்கின்றன என மேட்லாப் மற்றும் இதே போன்ற கட்டமைப்பைக் கொண்ட பிற தயாரிப்புகளின் பயனர்கள் கூறுகிறார்கள். அங்கு ஒரு பெரிய பலதரப்பட்ட சமூகம் அக்கருவியை நோண்டிப் பார்க்க (tinker) இயல்கிறது.



திறந்த மூல திட்டப் பங்களிப்பைத் தொடங்குவது எப்படி

முன்னதாக வெளியிடுக, அடிக்கடி வெளியிடுக

முன்னதாக வெளியிடுதல் மற்றும் அடிக்கடி வெளியிடுதல் லினக்ஸ் மேம்பாட்டு மாதிரியின் முக்கியமான அங்கமாகும். பெரும்பாலான நிரலாளர்கள் (நானும்தான்) மிகச்சிறியது தவிர மற்ற திட்டங்களுக்கு இது மோசமான கொள்கை என்று நம்பினர். ஏனெனில் தொடக்கப் பதிப்புகளில் வழு நிறைந்திருக்கும். உங்கள் பயனர்களின் பொறுமையை சோதிக்க விரும்ப மாட்டீர்கள்.

இக்கருத்துதான் பேராலயம்-பாணி வளர்ச்சிக்கான பொதுவான நம்பிக்கையை வலுப்படுத்தியது. பயனர்கள் முடிந்தவரை குறைந்த அளவு வழுக்களைப் பார்ப்பதே முக்கிய நோக்கமாக இருந்தால், நீங்கள் ஆறு மாதங்களுக்கு ஒரு முறையோ அல்லது அதைவிடக் குறைவாகவோதான் ஒரு பதிப்பை வெளியிடுவீர்கள். மேலும் வெளியீடுகளுக்கு இடையில் உள்ள காலத்தில் வழுத்திருத்தத்தில் அதிதீவிரமாகச் செயல்படுவீர்கள். ஈமாக்ஸ் சி கோர் (Emacs C core) இவ்வழியில் உருவாக்கப்பட்டது. லிஸ்ப் நிரலகம் (Lisp library) நடைமுறையில் இவ்வழியில் உருவாக்கப்படவில்லை.

ஏனென்றால் FSF (Free Software Foundation) கட்டுப்பாட்டிற்கு வெளியே செயலில் உள்ள லிஸ்ப்

காப்பகங்கள் இருந்தன. அங்கு எவரும் ஈமாக்ஸ் புதிய வெளியீடுகள் மற்றும் மேம்பாடு செய்து கொண்டிருக்கும் நிரல் பதிப்புகளைப் பதிவிறக்கம் செய்ய இயலும்.

இவற்றில் மிக முக்கியமானது, ஓஹியோ மாகாண ஈமாக்ஸ் லிஸ்ப் காப்பகம் (Ohio State Emacs Lisp archive). இன்றைய பெரிய லினக்ஸ் காப்பகங்களைப் போன்ற பல அம்சங்களை முன்செயல்படுத்தியது. ஆனால் நாங்கள் எவரும் என்ன செய்கிறோம் என்பதைப் பற்றியோ அல்லது அக்காப்பகம் இருப்பதால் FSF இன் பேராலயம்-பாணி மேம்பாட்டு மாதிரியில் எழும் சிக்கல்களைப் பற்றியோ மிகவும் கடினமாக யோசிக்கவில்லை. 1992 ஆம் ஆண்டில், அதிகாரப்பூர்வ ஈமாக்ஸ் லிஸ்ப் நிரலகத்தில் ஓஹியோ நிரலை முறையாக இணைக்க நான் ஒரு தீவிர முயற்சியை மேற்கொண்டேன். ஆனால் அரசியல் சிக்கல் காரணமாக நான் பெரும்பாலும் தோல்வியடைந்தேன்.

லினஸின் திறந்த வளர்ச்சிக் கொள்கை பேராலயம் பாணிக்கு முற்றிலும் எதிர் மாறானது

ஆனால் ஒரு வருடம் கழித்து, லினக்ஸ் பரவலாகத் தெரியவந்ததால், வித்தியாசமான மற்றும் மிகவும் ஆரோக்கியமான ஒன்று அங்கு நடக்கிறது என்பது தெளிவாகத் தெரிந்தது. லினஸின் திறந்த வளர்ச்சிக் கொள்கை பேராலயம் பாணிக்கு முற்றிலும் எதிர் மாறானது. லினக்ஸின் இணையக் காப்பகங்கள் வளர்ந்து வருகின்றன, பல விநியோகங்கள் (distros) வெளிவருகின்றன. மேலும் இவை அனைத்தும் இதுவரை கண்டிராத குறுகிய கால இடைவெளியில் கருநிரல் (kernel) வெளியீடுகளால் உந்தப்படுகின்றன.

லினஸ் தனது பயனர்களை இணை நிரலாளர்களாக மிகவும் பயனுள்ள முறையில் நடத்தினார்:

மணிமொழி 7. முன்னதாக வெளியிடுக. அடிக்கடி வெளியிடுக. மற்றும் வாடிக்கையாளர்களுக்கு செவிசாய்க்கவும்.

லினஸின் புதுப்புனைவு (innovation), பல பயனர் பின்னூட்டங்களை உள்ளடக்கி விரைவான வெளியீடுகளைச் செய்ததில் இல்லை (இது போன்ற ஒன்று நீண்ட காலமாக யூனிக்ஸ்-உலக பாரம்பரியமாக இருந்தது). ஆனால் தான் செய்யும் வேலையின் சிக்கலுக்குப் பொருந்தக்கூடிய தீவிரத்தன்மைக்கு அதை வளர்த்து விட்டதில்தான் (scaling it up). அந்த ஆரம்ப காலங்களில் (சுமார் 1991 இல்) ஒரு நாளைக்கு ஒரு முறைக்கு மேல் புதிய லினக்ஸ் கருநிரலை (Linux kernel) வெளியிடுவதும் நடக்காத சங்கதியல்ல! அவர் தனது இணை-நிரலாளர்களின் அடித்தளத்தை வளர்த்துக்கொண்டதால், கூட்டு முயற்சிக்கு மற்றவர்களை விடத் தீவிரமாக இணையத்தைப் பயன்படுத்தினார்.

ஆனால் இது எப்படி வேலை செய்தது? இது நான் காப்பியடிக்கக்கூடிய ஒன்றா அல்லது லினஸ் டோர்வால்ட்ஸின் சில தனித்துவமான மேதை குணங்களை நம்பியிருந்ததா?

நான் அப்படி நினைக்கவில்லை. லினஸ் ஒரு சிறந்த கொந்தர் என்பதில் ஐயமில்லை. நம்மில் எத்தனை பேர் ஒரு முழு உற்பத்தித்தரமான இயங்கு தளக் கருநிரலைப் புதிதாக உருவாக்க முடியும்? ஆனால் லினக்ஸ் எந்தவொரு அற்புதமான கருத்தியல் ரீதியான முன்னேற்றத்தையும் செயல்படுத்தவில்லை. ரிச்சர்ட் ஸ்டால்மேன் (Richard Stallman – NeWS) அல்லது ஜேம்ஸ் கோஸ்லிங் (James Gosling – ஜாவா) போன்று வடிவமைப்பில் லினஸ் ஒரு புதுப்புனைவு செய்த மேதை அல்ல (அதாவது

இதுவரை இல்லை). மாறாக, லினஸ் கணினிப் பொறியியலிலும் திட்டத்தைச் செயல்படுத்துவதிலும் ஒரு மேதையாக எனக்குத் தோன்றுகிறது. வழுக்கள் மற்றும் வளர்ச்சியின் முட்டுக்கட்டைகளைத் தவிர்ப்பதற்கான ஆறாவது அறிவு அவருக்கு இருந்தது. மற்றும் புள்ளி A இலிருந்து புள்ளி B வரையிலான குறைந்தபட்ச முயற்சிப் பாதையைக் கண்டுபிடிப்பதற்கான உண்மையான சாமர்த்தியம் இருந்தது. லினக்ஸின் முழு வடிவமைப்பும் இத்தரத்தை வெளிப்படுத்துகிறது மற்றும் லினஸின் அடிப்படையில் இடரைத் தவிர்க்கும் மற்றும் எளிமையாக்கும் வடிவமைப்பு அணுகுமுறையைப் பிரதிபலிக்கிறது.

எனவே, விரைவான வெளியீடுகளும், இணைய ஊடகத்தின் பயனை உச்சகட்டமாக உயர்த்துவதும் தற்செயலாக நடந்தவை அல்ல. ஆனால் குறைந்தபட்ச முயற்சிப்பாதை பற்றிய நுண்ணறிவு லினஸின் பொறியியல்-மேதையின் ஒருங்கிணைந்த பகுதி என்றால், அவர் எதை மிகுதியாக்கினார்? அவர் இந்த இயந்திரத்திலிருந்து எதைத் தயாரித்துத் தள்ளினார்?

இந்த முறையில் கேட்டால் கேள்விக்குத் தானாகவே பதில் கிடைக்கும். லினஸ் தனது கொந்தர்/பயனர்களைத் தொடர்ந்து ஊக்குவித்து, வெகுமதி அளித்து வந்தார். அவர்களின் தற்பெருமைக்கு (ego) திருப்தி அளிக்கும் வேலைகளில் வாய்ப்பளித்து ஆர்வமூட்டினார். தங்களின் வேலையில் தொடர்ந்த (தினமும் கூட) முன்னேற்றத்தைக் காண்பதன் மூலம் அவர்களுக்கு இந்த வெகுமதி தொடர்ந்து கிடைத்து வந்தது.

“அனேகம் பேர் கவனித்தால் அனைத்து வழுக்களும் எளியவையே” – லினஸின் விதி

லினஸ், வழுத்திருத்தம் மற்றும் மேம்பாட்டிற்கு

ஒதுக்கப்படும் நபர்-மணி நேரங்களின் எண்ணிக்கையை மிகுதியாக்குவதை நேரடியான இலக்காகக் கொண்டிருந்தார். நிரலில் உறுதியற்ற தன்மை மற்றும் ஏதேனும் தீர்வுகாண முடியாத தீவிரமான வழு இருந்தால் பயனர் அடித்தளம் முழுச்சோர்வு அடையக்கூடும் என்ற பிரச்சினை இருந்தும்கூட. லினஸ் நடந்துகொண்ட விதத்தைப் பார்த்தால் அவர் பின்வரும் மணிமொழியை நம்புவது போல் இருந்தது:

> மணிமொழி 8. போதுமான அளவு பீட்டா-சோதனையாளர் மற்றும் இணை-நிரலாளர் அடித்தளம் இருந்தால், அனேகமாக எந்தவொரு பிரச்சினையும் விரைவாக வகைப்படுத்தப்படும் மற்றும் எவராவது ஒருவருக்கு அதன் தீர்வு தெளிவாகத் தெரியும்.

அல்லது, சுருக்கமாக, “அனேகம் பேர் கவனித்தால் அனைத்து வழுக்களும் எளியவையே” இதை நான் “லினஸின் விதி” என்று அழைக்கிறேன்.

எனது முதல் உருவாக்கம் என்னவென்றால், ஒவ்வொரு பிரச்சினையும் “யாராவது ஒருவருக்கு வெளிப்படையானதாக இருக்கும்” என்றுதான்’. சிக்கலைப் புரிந்துகொண்டு சரிசெய்பவர்தான் அதை முதலில் வகைப்படுத்தியவர் என்று அவசியம் இல்லை, மேலும் வழக்கமாக அம்மாதிரி நடப்பதும் இல்லை என்று லினஸ் விளக்கம் கூறினார். “யாரோ ஒருவர் வழுவைக் கண்டுபிடிப்பார்,” என்று அவர் கூறுகிறார், “வேறொருவர் அதைப் புரிந்துகொண்டு தீர்வு காணுவார். இவற்றில் அதை முதலில் கண்டுபிடிப்பதுதான் பெரிய சவால் என்று என்னால் தீர்மானமாகச் சொல்ல முடியும்.” அந்த வழுத்திருத்தம் முக்கியம்தான், அது எப்படியென்று அதன் நடைமுறையை அடுத்த பகுதியில் இன்னும் விரிவாக

ஆராயும்போது பார்ப்போம். ஆனால் முக்கிய சங்கதி என்னவென்றால், இந்த செயல்முறையின் இரண்டு பகுதிகளும் (கண்டுபிடித்தல் மற்றும் தீர்வு காணுதல்) விரைவாக நடக்கின்றன.

பேராலயம்-பாணி மற்றும் சந்தை-பாணியின் அடிப்படை வேறுபாடு லினஸின் விதியில்தான் உள்ளது

லினஸின் விதியில்தான் பேராலயம்-பாணி மற்றும் சந்தை-பாணியின் அடிப்படையிலான முக்கிய வேறுபாடு உள்ளது என்று நான் நினைக்கிறேன். நிரலாக்கத்தின் பேராலயம்-பாணி பார்வையில், வழுக்கள் மற்றும் வளர்ச்சி பிரச்சினைகள் சிக்கலான, கேடார்ந்த, ஆழமான நிகழ்வுகளாகும். நீங்கள் அனைத்து வழுக்களையும் பிரித்தெடுத்து விட்டீர்கள் என்ற நம்பிக்கையை உருவாக்க அர்ப்பணிப்புள்ள சிலர் பல மாதங்கள் ஆய்வு செய்ய வேண்டும். இதனால் வெளியீடுகளுக்கு இடையில் நீண்ட இடைவெளிகள் இருக்கும். மற்றும் நீண்டகாலமாக எதிர்பார்த்த வெளியீடுகள் வழுவற்றதாக இல்லாதபோது தவிர்க்க முடியாத ஏமாற்றம் ஏற்படும்.

மறுபுறம், சந்தை பார்வையில், வழுக்கள் பொதுவாக ஆழமற்ற நிகழ்வுகள் என்று கருதலாம். அல்லது குறைந்தபட்சம், ஒவ்வொரு புதிய வெளியீட்டையும் துடிப்புள்ள ஆயிரக்கணக்கான இணை-நிரலாளர்கள் அக்கு வேறு ஆணி வேறாகப் பிரித்துப் பார்க்கும்போது அவை மிக விரைவாக ஆழமற்றதாக ஆகிவிடுகின்றன. ஆகவே, அதிக திருத்தங்களைப் பெறுவதற்காக நீங்கள் அடிக்கடி வெளியிடுகிறீர்கள். மேலும் நன்மை பயக்கும் பக்கவிளைவாக, வெளியீட்டில் எப்போதாவது ஒரு மோசமான வழு தப்பிச் சென்றால், எதிர்பார்ப்பு அதிகமில்லை என்பதால், உங்களுக்கு பாதிப்பும்

குறைவு.

லினக்ஸ் தொடர்ச்சியாகப் பல வருடங்கள் கூட மறுஇயக்கம் **(reboot)** செய்யாமல் இயங்க வல்லது

இவ்வளவுதான். இது போதும். லினஸின் விதி தவறானது என்றால், லினக்ஸ் கருநிரல் (Linux kernel) போன்ற சிக்கலான மற்றும் இத்தனை கைகளால் கொந்தப்பட்ட எந்த ஓர் அமைப்பும், எதிர்பாராத மோசமான செயல்விளைவுகள் மற்றும் கண்டுபிடிக்கப்படாத ஆழமான வழுக்களின் பாரம் தாங்காமல் ஒரு கட்டத்தில் உடைந்து விழுந்திருக்க வேண்டும். மாறாக லினஸின் விதி உண்மையாக இருந்தால், லினக்ஸில் ஏன் வழுக்கள் மிகக்குறைவாக உள்ளன மற்றும் லினக்ஸ் ஏன் தொடர்ச்சியாகப் பல மாதங்கள் அல்லது வருடங்கள் வரை மறுஇயக்கம் (reboot) செய்யாமல் இயங்க வல்லது என்பதை விளக்கப் போதுமானது.

ஒருவேளை இது அவ்வளவு ஆச்சரியமாக இருந்திருக்கக்கூடாது. சமூகவியலாளர்கள் பல ஆண்டுகளுக்கு முன்பு, சமமான நிபுணத்துவம் வாய்ந்த (அல்லது சமமாக அறியாமை கொண்ட) பார்வையாளர்களின் சராசரியான கருத்து, பார்வையாளர்களில் சீரற்ற முறையில் தேர்ந்தெடுக்கப்பட்ட (randomly-chosen) யாராவது ஒருவரின் கருத்தை விட, பெருமளவு நம்பகமான ஒரு முன்கணிப்பு என்று கண்டறிந்தனர். இதை டெல்ஃபி விளைவு (Delphi effect) என்று அழைத்தனர். இது ஓர் இயங்கு தளத்தின் வழுத்திருத்தத்திற்கு கூட பொருந்தும் என்று லினஸ் செய்து காட்டினார் என்று தோன்றுகிறது. அதாவது டெல்ஃபி விளைவு ஓர் இயங்கு தள கருநிரல் அளவு சிக்கலான வேலையில் கூட வளர்ச்சி சிக்கலைக் கட்டுப்படுத்தும்.

லினக்ஸ் சூழ்நிலையின் ஒரு சிறப்பு அம்சம், டெல்ஃபி விளைவுடன் கண்கூடாக உதவுகிறது.

எந்தவொரு திட்டத்திற்கும் பங்களிப்பாளர்கள் சுயமாகத் தேர்ந்தெடுக்கப்பட்டவர்கள் (தானாக உதவியளிக்க முன் வந்தவர்கள்). பங்களிப்புகள் சீரற்ற மாதிரியில் (random sample) இருந்து பெறப்படவில்லை என்று ஓர் ஆரம்ப விமர்சகர் சுட்டிக்காட்டினார். ஆனால் இவர்கள் மென்பொருளைப் பயன்படுத்த ஆர்வமுள்ளவர்கள், அது எவ்வாறு செயல்படுகிறது என்பதைப் பற்றி அறிந்துகொள்பவர்கள், இவர்கள் எதிர்கொள்ளும் பிரச்சனைகளுக்கு தீர்வு காண முயற்சி செய்பவர்கள், மேலும் ஓரளவு நல்ல தீர்வை உருவாக்குபவர்கள். இந்த வடிகட்டிகள் அனைத்தையும் கடந்து செல்லும் எவருக்கும் பயனுள்ள ஏதாவது பங்களிக்க வாய்ப்பு உள்ளது.

லினஸின் விதியை “வழுத்திருத்தம் இணையாக செய்யக்கூடியது **(parallelizable)”** என மாற்றிச் சொல்லலாம்

லினஸின் விதியை “வழுத்திருத்தம் இணையாக செய்யக்கூடியது” என மாற்றிச் சொல்லலாம். வழுத்திருத்தத்திற்கு சில ஒருங்கிணைப்பு நிரலாளர்களுடன் தொடர்பு கொள்ளத் தேவைப்பட்டாலும், வழுத்திருத்துபவர்களுக்கு இடையே குறிப்பிடத்தக்க தகவல் தொடர்பு தேவையில்லை. எனவே நிரலாளர்களைச் சேர்ப்பது தகவல் தொடர்பு பளு அதிகரிப்பு பிரச்சினையை உண்டாக்காது.

நடைமுறையில், வழுத்திருத்துபவர்கள் ஒருவர் செய்த வேலையை மற்றவர் திரும்பவும் வீணாகச் செய்வது லினக்ஸ் உலகில் ஒரு பிரச்சினையாகத் தெரியவில்லை. “முன்னதாக வெளியிடு, அடிக்கடி வெளியிடு” கொள்கையின் ஒரு விளைவு, பின்னூட்டங்களில் வந்த திருத்தங்களை விரைவாக வெளியிடுவதன் மூலம் இத்தகைய பயனற்ற

வேலைகளைக் குறைப்பதாகும்.

ப்ரூக்ஸ் (The Mythical Man-Month இன் ஆசிரியர்) இது தொடர்பான ஓர் அபிப்பிராயத்தைக் கூறினார்: “பரவலாகப் பயன்படுத்தப்படும் ஒரு மென்பொருளைப் பராமரிப்பதற்கான மொத்தச் செலவு பொதுவாக அதை உருவாக்குவதற்கான செலவில் 40 சதவிகிதம் அல்லது அதற்கு மேலும் ஆகும். ஆச்சரியப்படும் விதமாக, இந்தச் செலவு முக்கியமாக பயனர்களின் எண்ணிக்கையைப் பொருத்தது. அதிகப் பயனர்கள் அதிக வழுக்களைக் கண்டறிகின்றனர்.” [முக்கியத்துவம் சேர்க்கப்பட்டது].

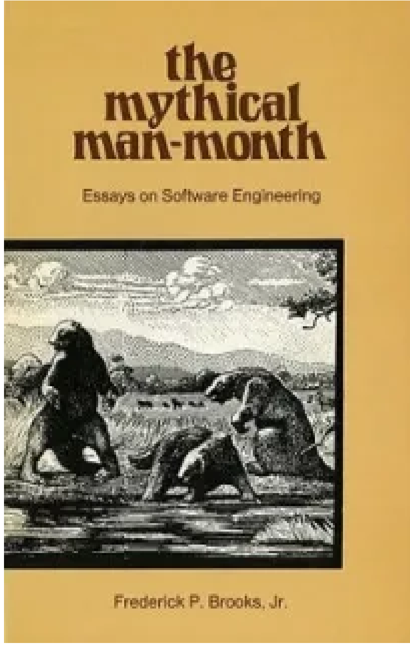
பயனர்கள் இணை நிரலாளர்களானால் பிரச்சினையை வெவ்வேறு கோணத்தில் அணுகுகின்றனர்

அதிகப் பயனர்கள் அதிக வழுக்களைக் கண்டறிகின்றனர், ஏனெனில் அதிகமான பயனர்களைச் சேர்ப்பது நிரலை பளுச்சோதனை செய்யும் பல்வேறு வழிகளைச் சேர்க்கிறது. பயனர்கள் இணை நிரலாளர்களாக இருக்கும்போது இந்த விளைவு பெருக்கப்படுகிறது. ஒவ்வொருவரும் வழு வகைப்படுத்தல் பணியை சற்று வித்தியாசமான உட்புரிதல் மற்றும் பகுப்பாய்வுக் கருவித்தொகுப்புடன், பிரச்சினையின் வெவ்வேறு கோணத்தில் அணுகுகின்றனர். இந்த மாறுபட்ட அணுகுமுறை காரணமாக “டெல்ஃபி விளைவு” துல்லியமாக வேலை செய்கிறது. வழுத்திருத்தத்தின் குறிப்பிட்ட சூழலில், இந்த மாறுபட்ட அணுகுமுறை வீண்வேலையைக் குறைக்கிறது.

எனவே அதிக பீட்டா-சோதனையாளர்களைச் சேர்ப்பது நிரலாளரின் பார்வையில் தற்போதைய “ஆழமான” வழுவின் சிக்கலைக் குறைக்காது. ஆனால் இது இந்த வழுவுடன் யாராவது ஒருவரின்

கருவித்தொகுப்பு பொருந்தக்கூடிய நிகழ்தகவை (probability) அதிகரிக்கிறது. அந்த நபருக்கு இந்த வழு ஆழமற்றதாக இருக்கும்.

கடுமையான வழுக்கள் இருந்தால் இழப்பிலிருந்து பாதுகாத்துக் கொள்ளவும் லினஸ் வழி ஏற்படுத்தியிருக்கிறார். வழுக்கள் நீக்கப்பட்ட லினக்ஸ் கருநிரல் பதிப்புகள் “நிலையானவை” என்று குறியிடப்பட்டிருக்கும். புதிய அம்சங்களைப் பெற விரும்பினால் பயனர்கள் புத்தம் புதிய பதிப்பைத் தேர்வு செய்யலாம். ஆனால் இதில் கடுமையான வழுக்கள் இருக்கக்கூடிய இடர் உண்டு என்று தெரியும். இந்த உத்தி இன்னும் பெரும்பாலான லினக்ஸ் கொந்தர்களால் முறையாகப் பின்பற்றப்படவில்லை, ஆனால் ஒருவேளை இதை மற்றவர்களும் பின்பற்றவேண்டும். இந்த இரண்டு தேர்வுகள் உள்ளன என்பது இரண்டையும் மிகவும் கவர்ச்சிகரமானதாக ஆக்குகிறது.



ஃப்ரெட் புரூக்ஸ் எழுதிய ஒரு நபரின் ஒரு மாத வேலை என்ற கட்டுக்கதை

எத்தனை பேர் கவனம் வைத்தால் சிக்கலை அடக்கியாள முடியும்

வழுத்திருத்தம் மற்றும் நிரல் வளர்ச்சியை சந்தை பாணி பெரிதும் துரிதப்படுத்துகிறது என்பதை மேம்போக்காக கவனிப்பது ஒன்று. தினசரி நிரலாளர் மற்றும் சோதனையாளர் நடத்தையின் நுண்மட்டத்தில் அது எப்படி, ஏன் வேலை செய்கிறது என்பதைத் துல்லியமாகப் புரிந்துகொள்வது மற்றொரு சங்கதி. இக்கட்டுரையில் (முதல் ஆய்வுக் கட்டுரைக்கு மூன்று ஆண்டுகளுக்குப் பிறகு எழுதப்பட்டது. அதைப் படித்து, தங்களின் சொந்த நடத்தையை மறுபரிசீலனை செய்த நிரலாளர்களின் நுண்ணறிவைப் பயன்படுத்தி) உண்மையான இயங்குமுறைகளை நாம் மிகக்கவனமாக ஆராய்வோம். தொழில்நுட்பத்தில் நாட்டமற்ற வாசகர்கள் இதை விடுத்து அடுத்த பகுதிக்குச் செல்லலாம்.

மூல-விழிப்புணர்வு இல்லாத **(non–source-aware)** பயனர்கள் வழுவை மீள்உருவாக்கும் செயல்படிகளைத் தருவதில்லை

மூல நிரல் தெரியாத பயனர்கள் சமர்ப்பிக்கும் வழு அறிக்கைகள் ஏன் மிகவும் பயனுள்ளதாக இருப்பதில்லை என்பதை சரியாகத்

தெரிந்துகொள்வது இதைப் புரிந்துகொள்வதற்கான ஒரு திறவுகோல். மூல-விழிப்புணர்வு இல்லாத பயனர்கள் மேற்பரப்பில் தென்படும் அறிகுறிகளைப் பற்றி மட்டுமே புகாரளிக்க முனைகிறார்கள். அவர்கள் தங்கள் சூழலை ஒரு பொருட்டாக எடுத்துக்கொள்வதில்லை. எனவே அவர்கள் முக்கியமான பின்னணித் தரவுகளைத் தருவதில்லை. மேலும் வழுவை மீண்டும் உருவாக்குவதற்கான நம்பகமான செயல்படிகளைத் தருவதும் அரிது.

சோதனையாளர் மற்றும் நிரலாளரின் செயலி பற்றிய மன மாதிரிகளுக்கு இடையே உள்ள பொருத்தமின்மையே இங்கு அடிப்படையான பிரச்சினையாகும். சோதனையாளர் செயலிக்கு வெளியிலிருந்து உள்ளே பார்க்கிறார், மாறாக நிரலாளர் செயலிக்கு உள்ளிருந்து வெளியே பார்க்கிறார். மூடிய-மூல வளர்ச்சியில் இவர்கள் இருவரும் இப்பாத்திரங்களில் (roles) மீள முடியாமல் சிக்கிக்கொண்டனர். ஆகவே ஒருவருக்கொருவர் புரியாமல் பேசி ஒருவரையொருவர் ஆழமாக வெறுப்படையச் செய்கிறார்கள்.

திறந்த மூல மேம்பாடு இத்தளையை அறுத்தெறிகிறது. இது உண்மையான மூல நிரலின் அடிப்படையில் பகிரும் மன மாதிரியை உருவாக்கவும், அதைப் பற்றி திறம்பட தொடர்புகொள்ளவும் சோதனையாளருக்கும் நிரலாளருக்கும் மிகவும் எளிதாக வழி செய்கிறது. நடைமுறையில், வெளிப்புறமாகத் தெரியும் அறிகுறிகளைப் புகாரளிக்கும் வழு அறிக்கையைவிட, மூல நிரல்-அடிப்படையிலான திட்டத்தின் மன மாதிரியை நேரடியாக இணைக்கும் வகையில் நிரலாளருக்கு மிகப்பெரிய ஆற்றலாதாயம் (leverage) உள்ளது.

மூல-நிரல் அளவில் வழு நிலைகளின் கோடி காட்டும் குணாதிசயத்தை வைத்தே புரிந்துகொள்ள இயலும்

பெரும்பாலான வழுக்கள், பெரும்பாலான நேரங்களில், மூல-நிரல் அளவில் அவற்றின் வழு நிலைகளின் ஓரளவு கோடி காட்டும் குணாதிசயத்தைக் கொடுத்தாலும் எளிதில் புரிந்துகொள்ள இயலும். உங்கள் பீட்டா-சோதனையாளர்களில் யாரேனும் ஒருவர், “இன்ன வரியில் எல்லைப் பிரச்சினை (boundary problem) உள்ளது” அல்லது “அ, ஆ மற்றும் இ நிபந்தனைகளின் கீழ், இந்த மாறி (variable) மீளமைகிறது” எனச் சுட்டிக்காட்டினால், பிரச்சினை தரும் நிரலை ஒருமுறைப் பார்ப்பதே அனேகமாகப் போதும். பிரச்சினையின் சரியான பயன்முறையைக் கண்டறிந்து ஒரு தீர்வை உருவாக்க இயலும்.

எனவே, பீட்டா-சோதனையாளர் அறிக்கைகள் மற்றும் முக்கிய நிரலாளர்(கள்) அறிந்தவற்றுக்கு இடையேயான தகவல்தொடர்பு மற்றும் கூட்டாற்றல் (synergy) இரண்டையும் மூல-நிரல் விழிப்புணர்வு பெரிதும் மேம்படுத்துகிறது. ஆகையால், பல கூட்டுப்பணியாளர்கள் இருந்தாலும் முக்கிய நிரலாளர்களின் நேரம் வீணாவதில்லை.

நிரலாளர் நேரத்தைப் பாதுகாக்கும் மற்றொரு சிறப்பியல்பு வழக்கமாகத் திறந்த மூல திட்டங்களிலுள்ள தொடர்பு அமைப்பு (communication structure) ஆகும். மேலே நான் “முக்கிய நிரலாளர்” என்று சொன்னேன். இது திட்ட மையத்திற்கும் (பொதுவாக ஒற்றை மைய நிரலாளர்தான் இருப்பார், மூன்று வரை இருக்கக்கூடும்) மற்றும் பீட்டா-சோதனையாளர்கள் மற்றும் பங்களிப்பாளர்களின் திட்ட வெளிவட்டம் (பெரும்பாலும் நூற்றுக்கணக்கான எண்ணிக்கையில் இருக்கலாம்)

ஆகியவற்றுக்கும் இடையேயான வேறுபாட்டைக் காட்டுகிறது.

மென்பொருள் திட்டம் தாமதமாகிறதே என்று கூடுதல் நிரலாளர்களைச் சேர்த்தால் அது மேலும் தாமதமாகும்

பாரம்பரிய மென்பொருள்-வளர்ச்சி அமைப்பு எதிர்கொள்ளும் அடிப்படைப் பிரச்சனை புரூக்ஸின் விதி: “மென்பொருள் திட்டம் தாமதமாகிறதே என்று கூடுதல் நிரலாளர்களைச் சேர்த்தால் அத்திட்டம் மேலும் தாமதமாகும்.” பொதுவாக ஒரு திட்டத்தின் சிக்கல்கள் மற்றும் தகவல்தொடர்பு பிரச்சினைகள் நிரலாளர்களின் எண்ணிக்கையின் வர்க்கத்தில் (square of the number of developers) அதிகரிக்கும் என்று புரூக்ஸின் விதி கணித்துள்ளது. ஆனால் செய்யும் வேலையோ நேர்கோட்டில் மட்டுமே உயரும்.

வெவ்வேறு நபர்களால் எழுதப்பட்ட நிரலுக்கு இடையே உள்ள இடைமுகங்களில் வழுக்கள் அதிகமாகக் காணப்படுகின்றன என்ற அனுபவத்தின் அடிப்படையில் புரூக்ஸின் விதி நிறுவப்பட்டது. மேலும் ஒரு திட்டத்தில் தகவல்தொடர்புகள் / ஒருங்கிணைத்தல் ஆகிய மேற்செலவுகள் நிரலாளர்களுக்கிடையிலான இடைமுகங்களின் எண்ணிக்கையுடன் உயரும். எனவே, நிரலாளர்களுக்கிடையிலான தகவல்தொடர்பு பாதைகளின் எண்ணிக்கையில் சிக்கல்கள் அதிகரிக்கின்றன. இது நிரலாளர்களின் எண்ணிக்கையின் வர்க்கமாக அளவிடப்படுகிறது (இன்னும் துல்லியமாக, N*(N – 1)/2 சூத்திரத்தின்படி N என்பது நிரலாளர்களின் எண்ணிக்கை). எடுத்துக்காட்டு: 50 நிரலாளர்களுக்கு 50 × (50 – 1)/2 = 1,225 தகவல் தொடர்பு அலைத்தடங்கள் (channels of communication) தேவைப்படும்.

புரூக்ஸின் விதி பகுப்பாய்வு (மேலும் வளர்ச்சிக்

குழுக்களில் அதிக எண்ணிக்கையில் நிரலாளர்களைச் சேர்ப்பது பற்றிய பயம்) ஒரு மறைவான தற்கோளை (hidden assumption) சார்ந்துள்ளது. அது திட்டத்தின் தகவல்தொடர்பு அமைப்புபடி எல்லோரும் மற்ற எல்லோருடனும் பேசுவார்கள் என்பதுதான். ஆனால் திறந்தமூல திட்டங்களில், திட்ட வெளிவட்டப் பங்களிப்பாளர்கள் பிரிக்கக்கூடிய இணையான துணைப் பணிகளில் (separable parallel subtasks) செயல்படுகிறார்கள். மற்றும் ஒருவருக்கொருவர் மிகக் குறைவாகவே தொடர்பு கொள்கிறார்கள். நிரல் மாற்றங்கள் மற்றும் வழு அறிக்கைகள் முக்கிய குழுவிற்குள் மட்டுமே கையாளப்படுகின்றன. மேலும் அச்சிறிய முக்கிய குழுவிற்குள் மட்டுமே நாம் முழு புரூக்ஸின் விதிக்கான மேற்செலவு செய்கிறோம்.

மூல-நிரல்-நிலையில் வழு அறிக்கை செயல்திறன் மிகுந்ததாக இருப்பதற்குப் பல காரணங்கள் உள்ளன

மூல-நிரல்-நிலையில் வழு அறிக்கை மிகவும் திறமையானதாக இருப்பதற்கு இன்னும் பல காரணங்கள் உள்ளன. பயனரின் பயன்பாட்டு முறை மற்றும் சூழலின் விவரங்களைப் பொறுத்து, ஒரு வழுவானது பல சாத்தியமான அறிகுறிகளைக் கொண்டிருக்கலாம் என்ற உண்மையை அவை மையமாகக் கொண்டுள்ளன. இத்தகைய பிழைகள் மிகவும் சிக்கலான மற்றும் நுட்பமான வழுக்கள் (இயங்குநிலை-நினைவகம்-மேலாண்மை (dynamic-memory-management) வழுக்கள் அல்லது நிர்ணயமற்ற நிறுத்த வாய்ப்பு (nondeterministic interrupt-window)) போன்றவை. இவற்றைத் தேவைக்கேற்ப மறுஉருவாக்கம் செய்வதும் அல்லது நிலையான பகுப்பாய்வு (static analysis) மூலம் கண்டறிவதும் கடினம். இவைதான் மென்பொருளில் நீண்ட கால பிரச்சினைகளை உருவாக்குகின்றன.

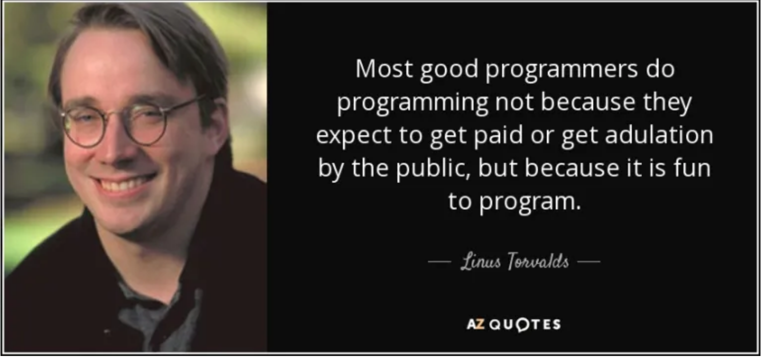
இதுபோன்ற பல-அறிகுறி வழுவின் தற்காலிக மூல-நிரல்-நிலை குணாதிசயத்தை சோதனையாளர் (எ.கா. “லைன் 1250 க்கு அருகில் சமிக்ஞை கையாளுதலில் ஒரு சாளரம் இருக்கலாம் போல் தோன்றுகிறது” அல்லது “அந்த இடையகத்தை (buffer) எங்கே சுழியமாக்குகிறீர்கள்?”) நிரலாளருக்குத் தெரிவிக்கலாம். நிரலில் சதா மூழ்கியிருக்கும் நிரலாளருக்கு பல்வேறு அறிகுறிகளுக்கான முக்கியமான துப்பு இதன்மூலம் கிடைக்கலாம். இது போன்ற சமயங்களில், எந்த வழுவின் காரணமாக வெளியில் தெரியும் பிரச்சினை ஏற்பட்டது என்பதை அறிவது கடினமாக இருக்கலாம். ஆனால் அடிக்கடி வெளியிடும் போது, அதை அறிவது தேவையற்றது. மற்ற கூட்டுப்பணியாளர்கள் தங்கள் வழு சரி செய்யப்பட்டதா இல்லையா என்பதை விரைவாகக் கண்டறியும் வாய்ப்பு உள்ளது. பல சந்தர்ப்பங்களில், மூல-நிலை வழு அறிக்கைகள் மூலம் எந்தக் குறிப்பிட்ட வழுத்திருத்தத்தில் என்று தெரியாமலே சில பிரச்சினைகள் தீர்ந்துவிடும்.

சிக்கலான பல-அறிகுறி பிழைகள் மேற்பரப்பு அறிகுறிகளிலிருந்து உண்மையான வழு வரை பல தடயப் பாதைகளைக் கொண்டிருக்கும். ஒரு நிரலாளர் அல்லது சோதனையாளர் பின்பற்றக்கூடிய பாதை எது என்பது அந்த நபரின் சுற்றுச்சூழலின் நுணுக்கங்களைப் பொறுத்து இருக்கலாம். மேலும் அது காலப்போக்கில் வெளிப்படையாகத் தீர்மானிக்க முடியாத வகையில் மாறலாம். இதன் விளைவாக, ஒவ்வொரு நிரலாளர் மற்றும் சோதனையாளர் ஓர் அறிகுறியின் காரணத்தைத் தேடும் போது திட்டத்தின் நிலை இடத்தின் ஒரு அரை-சீரற்ற (semi-random) மாதிரியைத்தான் பார்க்க இயலும். மிகவும் நுட்பமான மற்றும் சிக்கலான வழு, அந்த மாதிரியின் பொருத்தத்திற்கு உத்தரவாதம் அளிக்கும் திறன் குறைவாக இருக்கும்.

எளிமையான மற்றும் எளிதில் மீள்உருவாக்கக்கூடிய வழுக்களுக்கு, முக்கியத்துவம் “சீரற்றது (random)” என்பதை விட “அரை (semi)” யில் இருக்கும். வழுத்திருத்த திறன் மற்றும் நிரல் கட்டமைப்பு ஆகியவற்றுடன் நெருக்கம் மிகவும் முக்கியமானது. ஆனால் சிக்கலான வழுக்களுக்கு, முக்கியத்துவம் “சீரற்றது” என்பதில் இருக்கும். இச்சூழ்நிலையில், பலர் தடயங்களைப் பின்தொடர்வது ஒரு சிலர் மட்டுமே தடயங்களைப் பின்தொடர்வதை விட மிகவும் பயனுள்ளதாக இருக்கும். அந்த சிலருக்கு சராசரி திறன் நிலை அதிகம் இருந்தாலும் கூட.

தடயப் பாதைகளைப் **(trace paths)** பின்தொடரப் பலர் இணையாக முயற்சி செய்வதால் வழு நீக்கல் எளிதாகிறது

வெவ்வேறு மேற்பரப்பு அறிகுறிகளிலிருந்து ஒரு வழு வரை தடயப் பாதைகளைப் பின்தொடர்வதில் உள்ள சிரமம், அறிகுறிகளைப் பார்த்து கணிக்க முடியாத வகையில் கணிசமாக வேறுபடும் பட்சத்தில், இந்த விளைவு மேலும் பெருகும். ஒரு நிரலாளர் இப்பாதைகளை ஒன்றன்பின் ஒன்றாக எடுத்துக்கொண்டால், முதல் முயற்சியிலேயே கடினமான தடயப் பாதையை தேர்ந்தெடுக்கும் வாய்ப்பு உண்டு. மறுபுறம், விரைவான வெளியீடுகளைச் செய்யும்போது பலர் இப்பாதைகளை இணையாக முயற்சி செய்கிறார்கள் என்று வைத்துக்கொள்வோம். அவர்களில் ஒருவர் உடனடியாக எளிதான பாதையைக் கண்டுபிடித்து, மிகக் குறுகிய காலத்தில் வழுவை அகற்றுவார். திட்டப் பராமரிப்பாளர் அதைப் பார்த்து ஒரு புதிய வெளியீட்டை அனுப்புவார். அதே வழுவில் வேலை செய்யும் மற்றவர்கள் தங்கள் கடினமான தடயப் பாதைகளில் அதிக நேரம் செலவழிக்கும் முன் நிறுத்த முடியும்.



லினக்ஸ் மற்றும் கிட் உருவாக்குனர் லினஸ் டோர்வால்ட்ஸ் கூற்று

ரோஜா எப்போது ரோஜா அல்ல?

லினஸின் செயல்முறையை நுட்பமாக ஆய்வு செய்து, அது ஏன் வெற்றிகரமாக இருந்தது என்பது பற்றிய ஒரு கோட்பாட்டை உருவாக்கினேன். பிறகு, எனது புதிய திட்டத்தில் (லினக்ஸ் அளவுக்குச் சிக்கலானதோ அல்லது பெரிய அளவிலானதோ இல்லை என்றாலும் கூட) இக்கோட்பாட்டைச் சோதிக்கத் தெரிந்தே ஒரு முடிவை எடுத்தேன்.

நிரலை மையமாகவும், தரவுக் கட்டமைப்புகளை நிரலுக்கு ஆதரவாகவும் கருதினார்

ஆனால் நான் செய்த முதல் வேலை, பாப்கிளையன்டை மறுசீரமைத்து எளிமைப்படுத்துவதுதான். கார்ல் ஹாரிஸின் செயலாக்கம் மிகவும் நன்றாக இருந்தது, ஆனால் பல C நிரலாளர்களுக்குப் பொதுவான தேவையற்ற சிக்கலை வெளிப்படுத்தியது. அவர் நிரலை மையமாகவும், தரவுக் கட்டமைப்புகளை நிரலுக்கு ஆதரவாகவும் கருதினார். இதன் விளைவாக, நிரல் அழகாக இருந்தது, ஆனால் தரவுக் கட்டமைப்பு தற்காலிகமாகவும் அசிங்கமாகவும் இருந்தது (அதாவது இந்த மூத்த LISP கொந்தரின் உயர் தரத்தின்படி).

இருப்பினும், நிரல் மற்றும் தரவுக் கட்டமைப்பு வடிவமைப்பை மேம்படுத்துவதைத் தவிர,

மாற்றியெழுதுவதற்கு எனக்கு மற்றொரு நோக்கம் இருந்தது. நான் முழுமையாகப் புரிந்து கொண்ட ஒன்றாக அதைப் பரிணமிக்க வேண்டும். உங்களுக்குப் புரியாத திட்டத்தில் வழுக்களைச் சரிசெய்வதற்குப் பொறுப்பேற்பதில் ஏது மகிழ்ச்சி.

புத்திசாலித்தனம் உள்ள தரவுக் கட்டமைப்புகளும் புத்திசாலித்தனம் இல்லாத நிரலும்

சுமார் ஒரு மாதம் வரை, நான் கார்லின் அடிப்படை வடிவமைப்பின் தாக்கங்களை மட்டுமே பின்பற்றினேன். நான் செய்த முதல் தீவிர மாற்றம் IMAP ஆதரவைச் சேர்ப்பதாகும். நெறிமுறை இயந்திரங்களை (protocol machines) ஒரு பொதுவான இயக்கி மற்றும் மூன்று வழிமுறை அட்டவணைகளாக (POP2, POP3 மற்றும் IMAP க்கு) மறுசீரமைப்பதன் மூலம் இதைச் செய்தேன். இதுவும் முந்தைய மாற்றங்களும் நிரலாளர்கள் மனதில் கொள்ள வேண்டிய பொதுவான ஒரு கொள்கையை விளக்குகின்றன, குறிப்பாக இயற்கையாகவே இயங்குநிலை வகைப்படுத்தல் (dynamic typing) செய்யாத C போன்ற மொழிகளில்:

> மணிமொழி 9. புத்திசாலித்தனம் உள்ள தரவுக் கட்டமைப்புகளும் புத்திசாலித்தனம் இல்லாத நிரலும், இதற்கு நேர் மாறாக இருப்பதைவிட, மிகவும் சிறப்பாக வேலை செய்கிறது.

ப்ரூக்ஸ், அத்தியாயம் 9: “உங்கள் பாய்வு விளக்கப்படத்தைக் காட்டுங்கள் ஆனால் உங்கள் அட்டவணைகளை மறைத்து விடுங்கள், எனக்குத் தொடர்ந்து மர்மமாகவே இருக்கும். உங்கள் அட்டவணைகளை எனக்குக் காட்டுங்கள், பொதுவாக உங்கள் பாய்வு விளக்கப்படம் எனக்குத் தேவைப்படாது, அது தெளிவாகத்தான் இருக்கும்.” முப்பது வருடக் கலைச்சொற்கள்/கலாச்சார

மாற்றத்தைக் கணக்கில் கொண்டு பார்த்தால், இதுவும் அதே கருத்துதான்.

இந்தக் கட்டத்தில் (செப்டம்பர் 1996, தொடங்கியதிலிருந்து சுமார் ஆறு வாரங்கள்) ஒரு பெயர் மாற்றம் செய்தால் நன்றாக இருக்கும் என்று நான் நினைக்க ஆரம்பித்தேன். எல்லாவற்றிற்கும் மேலாக, இது ஒரு POP பயனர் செயலி மட்டும் அல்ல. ஆனால் நான் தயங்கினேன், ஏனென்றால் வடிவமைப்பில் இதுவரை உண்மையிலேயே புதிதாக எதுவும் இல்லை. எனது பாப்கிளையன்ட் பதிப்பு இன்னும் தன் சொந்த அடையாளத்தை உருவாக்கவில்லை.

டோர்வால்ட்ஸ் சரியாகச் செய்ததைப் பற்றிய எனது கோட்பாட்டை எப்படிச் சோதித்தேன்

பெறப்பட்ட அஞ்சலை SMTP போர்ட்டுக்கு எவ்வாறு அனுப்புவது என்பதை பாப்க்ளையன்ட் கற்றுக்கொண்டபோது அந்த நிலை தலைகீழாக மாறியது. நாம் சிறிது நேரத்தில் அதற்கு வருவோம். ஆனால் முதலில், லினஸ் டோர்வால்ட்ஸ் சரியாகச் செய்ததைப் பற்றிய எனது கோட்பாட்டைச் சோதிக்க இத்திட்டத்தைப் பயன்படுத்த முடிவு செய்தேன் என்று நான் முன்பே சொன்னேன் அல்லவா. நான் அதை எப்படிச் செய்தேன் என்று நீங்கள் தாராளமாகக் கேட்கலாம்? இதோ இந்த வழிகளில்:

- நான் முன்னதாகவும், அடிக்கடியும் வெளியிட்டேன் (ஒவ்வொரு பத்து நாட்களுக்குள்; தீவிர வளர்ச்சியின் போது, ஒரு நாளைக்கு ஒரு முறை).

- ஃபெட்ச்மெயில் பற்றி என்னைத் தொடர்பு கொண்ட அனைவரையும் சேர்த்து எனது பீட்டா பட்டியலை வளர்த்தேன்.

- நான் வெளியிடும் போதெல்லாம் பீட்டா பட்டியலுக்கு பயனர்களைப் பங்கேற்க ஊக்குவிக்கும் வகையில் அரட்டையான அறிவிப்புகளை அனுப்பினேன்.

- எனது பீட்டா சோதனையாளர்களின் கருத்துகளுக்குச் செவி சாய்த்தேன். வடிவமைப்பு முடிவுகளைப் பற்றி அவர்களிடம் கருத்துக் கேட்டேன் மற்றும் அவர்கள் ஒட்டு நிரல்கள் (patch) மற்றும் பின்னூட்டங்களை அனுப்பும் போதெல்லாம் அவர்களைத் தட்டிக் கொடுத்து ஊக்கமளித்தேன்.

இந்த எளிய நடவடிக்கைகள் கைமேல் பலன் அளித்தன

இந்த எளிய நடவடிக்கைகளின் பலன் உடனடியாக இருந்தது. திட்டத்தின் தொடக்கத்திலிருந்தே, மற்ற நிரலாளர்கள் கிடைக்காதா என்று ஏங்கும் தரத்தில், வழு அறிக்கைகளைப் பெற்றேன். அவற்றுடன் பெரும்பாலும் நல்ல வழுத்திருத்தங்களும் இணைக்கப்பட்டிருந்தன. நான் சிந்தனைமிகுந்த விமர்சனங்களைப் பெற்றேன். எனக்கு ரசிகர் அஞ்சல்கள் வந்தன. அறிவார்ந்த அம்சப் பரிந்துரைகளைப் பெற்றேன். இவையெல்லாம் அடுத்த மணிமொழிக்கு வழிவகுக்கின்றன:

> மணிமொழி 10. உங்கள் பீட்டா-சோதனையாளர்களை உங்களின் மிகவும் மதிப்புமிக்க வளங்களாக நீங்கள் கருதினால், அவர்கள் உங்களின் மிகவும் மதிப்புமிக்க வளங்களாகவே ஆகிவிடுவார்கள்.

ஃபெட்ச்மெயிலின் வெற்றிக்கான ஒரு சுவாரசியமான அளவீடு திட்ட பீட்டா பட்டியலின் (fetchmail-friends)

பெரிய அளவு. இந்தக் கட்டுரைத் தொகுப்பின் சமீபத்திய திருத்தத்தின் போது (நவம்பர் 2000) இது 287 உறுப்பினர்களைக் கொண்டிருந்தது மற்றும் வாரத்திற்கு இரண்டு அல்லது மூன்று பேரைச் சேர்க்கிறது.

உண்மையில், மே 1997 இன் பிற்பகுதியில் நான் திருத்தியபோது, ஒரு சுவாரசியமான காரணத்திற்காக பட்டியல் அதன் அதிகபட்சமான 300 உறுப்பினர்களிலிருந்து இழக்கத் தொடங்கியதைக் கண்டேன். ஃபெட்ச்மெயில் அவர்களுக்கு நன்றாக வேலை செய்வதால், இனி பட்டியல் அஞ்சல்களைப் பார்க்க வேண்டிய அவசியமில்லை என்பதால், பலர் தங்களைக் குழுவிலிருந்து விலக்கச் சொன்னார்கள்! ஒருவேளை இது ஒரு முதிர்ந்த சந்தை-பாணி திட்டத்தின் இயல்பான வாழ்க்கைச் சுழற்சியின் (life-cycle) ஒரு பகுதியாக இருக்கலாம்.













ஃபெட்ச்மெயில் பதிவிறக்க பக்கம்

பாப்கிளையன்ட் ஃபெட்ச்மெயில் ஆகிறது

SMTP போர்ட்டுக்கு அஞ்சலை அனுப்பும் அம்சம் – ஒரு பயனர் கொடுத்த அற்புதமான யோசனை

பயனர் கணினியின் SMTP போர்ட்டுக்கு அஞ்சலை அனுப்புவதற்காக ஹாரி ஹோச்ஹெய்சர் (Harry Hochheiser) தனது துண்டு நிரலை எனக்கு அனுப்பியதுதான் திட்டத்தில் உண்மையான திருப்புமுனையாக அமைந்தது. இந்த அம்சத்தை நம்பகமான முறையில் செயல்படுத்தினால், மற்ற எல்லா அஞ்சல் கொண்டு சேர்க்கும் முறைகளும் வழக்கற்றுப் போய்விடும் என்பதை நான் உடனடியாக உணர்ந்தேன்.

பல வாரங்களாக நான் ஃபெட்ச்மெயிலை கொஞ்சம் கொஞ்சமாக மேம்பாடு செய்து கொண்டிருந்தேன். அதே நேரத்தில் இடைமுக வடிவமைப்பு வேலை செய்யக்கூடியதாக இருந்தது, ஆனால் நேர்த்தியாக இல்லை. மற்றும் பல அபத்தமான விருப்பமைவுகளுடன் இருந்தது. கொண்டு வந்த அஞ்சலை அஞ்சல் பெட்டிக் கோப்பு (mailbox file) அல்லது நிலையான வெளியீட்டிற்கு (standard output) அனுப்புவதற்கான விருப்பங்கள் என்னை மிகவும் தொந்தரவு செய்தன, ஆனால் ஏன் என்று எனக்குப் புரியவில்லை.

(இணைய அஞ்சலின் விவரமான

தொழில்நுட்பத்தில் உங்களுக்கு நாட்டம் இல்லாவிட்டால், அடுத்த இரண்டு பத்திகளைத் தவிர்க்கலாம்.)

SMTP மேலனுப்புதல் பற்றி யோசித்தபோது நான் பார்த்தது என்னவென்றால், பாப்க்ளையன்ட் பல வேலைகளைச் செய்ய முயற்சி செய்கிறது. இது ஒரு அஞ்சல் போக்குவரத்து முகவராகவும் (mail transport agent – MTA) மற்றும் அஞ்சல் விநியோக முகவராகவும் (mail delivery agent – MDA) வடிவமைக்கப்பட்டுள்ளது. SMTP மேலனுப்புதல் மூலம், அது MDA வேலையைத் தவிர்த்து ஒரு தூய MTA ஆக இருக்கலாம். செண்ட்மெயில் (sendmail) செய்வது போலவே உள்ளூர் வினியோகத்திற்கு மற்ற செயலிகளுக்கு அஞ்சலைக் கொடுத்து விடலாம்.

TCP/IP ஆதரவுள்ள எந்த இயங்குதளத்திலும் போர்ட் 25 இருக்கும் என கிட்டத்தட்ட உத்தரவாதம் அளிக்கப்பட்டிருக்கும் போது, அஞ்சல் விநியோக முகவரை உள்ளமைப்பது அல்லது அஞ்சல் பெட்டியைப் பூட்டிவிட்டு பின்னிணைப்பு செய்வது போன்ற அனைத்து சிக்கல்களையும் ஏன் இழுத்துப் போட்டுக் கொள்ள வேண்டும்? குறிப்பாக, கொண்டு வந்த அஞ்சல் சாதாரண அனுப்புநர்-தொடங்கிய SMTP அஞ்சலைப் போல் இருக்கும் என்று உத்தரவாதம் அளிக்கப்படும்போது. இதுதானே நாம் உண்மையில் விரும்புவது.

(மீண்டும் உயர் நிலைக்கு….)

லினஸின் முறைகளை மனப்பூர்வமாகப் பின்பற்ற முயற்சித்ததன் மூலம் நான் பெற்ற மிகப்பெரிய பலன்

முந்தைய தொழில்நுட்பச் சொற்கள் உங்களுக்குப் புரியாவிட்டாலும், இங்கு பல முக்கியமான பாடங்கள் உள்ளன. முதலாவதாக, இந்த SMTP-மேலனுப்புதல்

கருத்து லினஸின் முறைகளை மனப்பூர்வமாகப் பின்பற்ற முயற்சித்ததன் மூலம் நான் பெற்ற மிகப்பெரிய ஒற்றைப் பலன் ஆகும். ஒரு பயனர் எனக்கு இந்த அற்புதமான யோசனையை வழங்கினார். நான் செய்ய வேண்டியதெல்லாம் அதன் தாக்கங்களைப் புரிந்துகொள்வதுதான்.

> மணிமொழி 11. உங்களுக்கு நல்ல யோசனைகள் தோன்றுவதற்கு அடுத்து சிறந்தது உங்கள் பயனர்களின் நல்ல யோசனைகளை அடையாளம் காண்பதே. சில நேரங்களில் பிந்தையதே மேலும் சிறந்தது.

நீங்கள் உண்மையிலேயே அடக்கமாக இருந்தால், கண்டுபிடிப்பை நீங்களே செய்தீர்கள் என உலகம் முழுவதும் நினைப்பார்கள்

சுவாரசியமாக, மற்றவர்களுக்கு எவ்வளவு கடன்பட்டிருக்கிறீர்கள் என்பதில் நீங்கள் முழுமையாக உண்மையிலேயே அடக்கமாகவும் இருந்தால், கண்டுபிடிப்பின் ஒவ்வொரு பகுதியையும் நீங்களே செய்தீர்கள் எனவும் உங்கள் உள்ளார்ந்த மேதையைப் பற்றி அடக்கமாக இருக்கிறீர்கள் எனவும் உலகம் முழுவதும் நினைப்பார்கள். லினஸுக்கு இது எவ்வளவு நன்றாக வேலை செய்தது என்பதை நாம் அனைவரும் பார்க்கலாம்!

(ஆகஸ்ட் 1997 இல் நடந்த முதல் பெர்ல் (Perl) மாநாட்டில் நான் எனது உரையை நிகழ்த்தியபோது, அபாரமான கொந்தர் லாரி வால் (Larry Wall – பெர்ல் நிரல் மொழி உருவாக்கியவர்) முன் வரிசையில் இருந்தார். மேலே கடைசி வரிக்கு வந்ததும், அவர் சமயப் போதனை பாணியில், “சொல்லுங்கள், சொல்லுங்கள், சகோதரரே!” என்றார். பார்வையாளர்கள் அனைவரும் சிரித்தனர். ஏனென்றால் இது பெர்ல் நிரல் மொழி

கண்டுபிடிப்பாளருக்கும் வேலை செய்தது என்பது அவர்களுக்குத் தெரியும்.)

சில வாரங்களுக்குப் பிறகு அதே உற்சாகத்தில் திட்டத்தை இயக்கிய பிறகு, எனது பயனர்களிடமிருந்து மட்டுமல்ல, இதைக் கேள்விப்பட்ட மற்றவர்களிடமிருந்தும் இதே போன்ற பாராட்டுகளைப் பெற ஆரம்பித்தேன். அந்த மின்னஞ்சலில் சிலவற்றை சேமித்து வைத்துள்ளேன். எப்போதாவது என் வாழ்க்கை பயனுள்ளதா என்று நான் யோசிக்க ஆரம்பித்தால் அதை எடுத்து மீண்டும் பார்ப்பேன் :-).

இதுநாள்வரைத் தவறான சிக்கலைத் தீர்க்க முயற்சித்து வந்திருக்கிறேன்

ஆனால் எல்லா வகையான வடிவமைப்பிற்கும் பொதுவான இரண்டு அடிப்படையான, அரசியல் அல்லாத பாடங்கள் இங்கே உள்ளன.

> மணிமொழி 12. பல நேரங்களில், பிரச்சினை இன்னது என்ற உங்கள் கருத்து தவறானது என்பதை உணரத் தொடங்குவதில் இருந்தே மிகவும் குறிப்பிடத்தக்க மற்றும் புதுமையான தீர்வுகள் உருவாகின்றன.

அனைத்து வகையான அறைகுறை உள்ளூர் விநியோக முறைகளுடன் ஒருங்கிணைந்த MTA/MDA வாக பாப்கிளையண்டைத் தொடர்ந்து உருவாக்குவதன் மூலம் தவறான சிக்கலைத் தீர்க்க முயற்சித்து வந்திருக்கிறேன். ஃபெட்ச்மெயிலின் வடிவமைப்பானது சாதாரண SMTP-பேசும் இணைய அஞ்சல் பாதையின் ஒரு பகுதியான ஒரு தூய MTA என அடிப்படையிலிருந்து மறுபரிசீலனை செய்யப்பட வேண்டும்.

வளர்ச்சியில் நீங்கள் ஒரு சுவர் போன்ற தடையில் முட்டி நிற்கும் போது – அடுத்த ஒட்டு நிரலைத் தாண்டிச் சிந்திக்க உங்களுக்குக் கடினமாக இருக்கும்போது – உங்களுக்குச் சரியான பதில் கிடைத்ததா என்று கேட்காமல், சரியான கேள்வியைத்தான் கேட்கிறீர்களா என்று பார்க்க வேண்டிய நேரம் இது. ஒருவேளை பிரச்சினையையே மாற்றியமைக்க வேண்டியிருக்கலாம்.

செயல்திறனை இழக்காமல் செய்ய முடிந்தால், பயனற்ற பழைய அம்சங்களை நீக்கத் தயங்காதீர்கள்

ஆகவே, நான் என் பிரச்சினையை மாற்றியமைத்தேன். தெளிவாக, செய்ய வேண்டியது இதுதான் (1) SMTP மேலனுப்புதல் ஆதரவைப் பொதுவான இயக்கியில் (generic driver) மாற்றி எழுதுவது, (2) அதை இயல்புநிலை பயன்முறையாக மாற்றுவது மற்றும் (3) இறுதியில் மற்ற எல்லா கொண்டு சேர்க்கும் முறைகளையும், குறிப்பாக கோப்பில் கொண்டு சேர் (deliver-to-file) மற்றும் நிலையான வெளியீட்டில் கொண்டு சேர் (deliver-to-standard-output) ஆகியவற்றை நீக்குவது.

நான் மூன்றாம் படியைப் பற்றிக் கொஞ்ச காலம் தயங்கினேன். மாற்று கொண்டு சேர்க்கும் வழிமுறைகளைச் சார்ந்திருக்கும் நீண்ட கால பாப்கிளையன்ட் பயனர்களை எரிச்சல்படுத்தப் பயந்தேன். கோட்பாட்டின்படி, அதே விளைவுகளைப் பெற அவர்கள் உடனடியாக .forward கோப்புகள் அல்லது சமமான செண்ட்மெயில் அல்லாத அஞ்சல்களுக்கு மாறலாம். ஆனால் நடைமுறையில் இந்த மாற்றம் குழப்பமாக இருக்கக்கூடும்.

ஆனால் நான் அதைச் செய்தபோது, பலன்கள் மிகப் பெரிதாக இருந்தன. இயக்கி நிரலின் மிக மோசமான பகுதிகள் மறைந்துவிட்டன. உள்ளமைவு அடியோடு எளிமையாகிவிட்டது. திட்டத்தின் MDA

மற்றும் பயனரின் அஞ்சல்பெட்டி கிடைக்குமா என்று அலைய வேண்டாம், கோப்பு பூட்டுதலை இயங்குதளம் ஆதரிக்கிறதா என்பதைப் பற்றியும் கவலைப்பட வேண்டாம்.

மேலும், அஞ்சலை இழக்கக்கூடிய ஒரே வழியும் மறைந்துவிட்டது. நீங்கள் ஒரு கோப்பிற்குக் கொண்டு சேர்க்கக் குறிப்பிட்டு, வட்டு (disk) நிரம்பியிருந்தால், உங்கள் அஞ்சல் தொலைந்துவிடும். SMTP மேலனுப்புதலில் அஞ்சல் தொலைந்து போகாது. ஏனெனில் செய்தியைக் கொண்டு சேர்க்க முடியாவிட்டால் அல்லது குறைந்த பட்சம் பின்னர் கொண்டு சேர்ப்பதற்காக சுருட்டி வைக்காவிட்டால் (spooled) உங்கள் SMTP கேட்குநர் (listener) OK அனுப்பாது.

மேலும் செயல்திறன் மேம்பட்டது (ஒரே ஓட்டத்தில் உங்களுக்குத் தெரியவராது என்றாலும்). இந்த மாற்றத்தின் மற்றொரு முக்கிய நன்மை என்னவென்றால், கையேடு மிகவும் எளிமையாகிவிட்டது.

பின்னர், இயங்குநிலை SLIP சம்பந்தப்பட்ட சில தெளிவற்ற சூழ்நிலைகளைக் கையாள அனுமதிப்பதற்காக, பயனர் குறிப்பிட்ட உள்ளூர் MDA மூலம் கொண்டுசேர்த்தலை மீண்டும் கொண்டு வர வேண்டியிருந்தது. ஆனால் நான் அதைச் செய்வதற்கான ஓர் எளிய வழியைக் கண்டுபிடித்தேன்.

இதன்மூலம் நாம் கற்க வேண்டிய பாடம் என்ன? செயல்திறனை இழக்காமல் உங்களால் செய்ய முடியும் என்றால், பயனற்றுப்போன பழைய அம்சங்களைத் தூக்கி எறியத் தயங்காதீர்கள். அன்டோயின் டி செயின்ட்-எக்சூபெரி (Antoine de Saint-Exupéry – அவர் புகழ்பெற்ற குழந்தைகள் புத்தகங்களை எழுதாத போது விமானியாகவும்

விமான வடிவமைப்பாளராகவும் இருந்தார்) கூறினார்:

> மணிமொழி 13. “(வடிவமைப்பில்) முழுமை அடைவது என்பது சேர்ப்பதற்கு எதுவும் இல்லாதபோது அல்ல, மாறாக நீக்குவதற்கு எதுவும் இல்லாதபோதுதான்.”

உங்கள் நிரல் சிறப்பாகவும் எளிமையாகவும் இருக்கும் போது, அது சரியானது என்று உங்களுக்குத் தெரியவரும். இவ்வாறாக, ஃபெட்ச்மெயில் வடிவமைப்பு அதன் முன்னோர் பாப்கிளையன்டிலிருந்து வேறுபட்ட தன் சொந்த அடையாளத்தைப் பெற்றது.

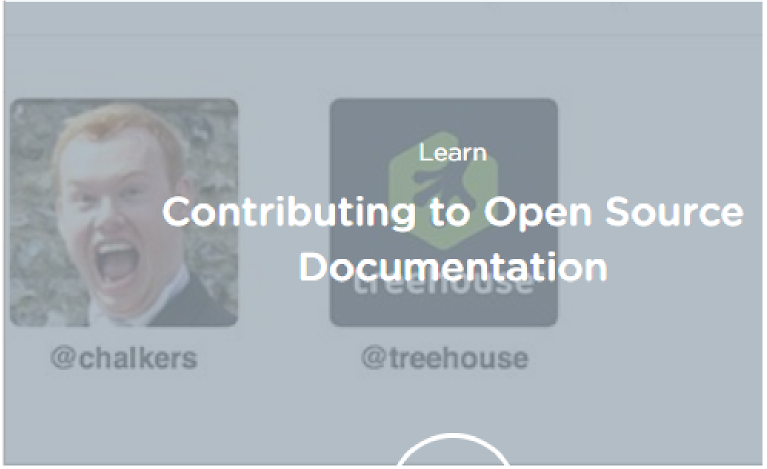
ஆகவே பெயர் மாற்றும் நேரம் வந்துவிட்டது. புதிய வடிவமைப்பு பழைய பாப்கிளையன்டை விட செண்ட்மெயிலின் இரட்டையர் போன்றது. இரண்டும் MTA க்கள், ஆனால் செண்ட்மெயில் தள்ளிக் கொண்டுசேர்க்கிறது, புதிய பாப்கிளையன்ட் இழுத்துக் கொண்டுசேர்க்கிறது. எனவே, தொடங்கி இரண்டே மாதங்களில், நான் அதை ஃபெட்ச்மெயில் (fetchmail) என மறுபெயரிட்டேன்.

வழுத்திருத்தம் போல மேம்பாடும் வடிவமைப்பும் கூட இணையாக செய்யக்கூடியவையே **(parallelizable)**

SMTP கொண்டுசேர்ப்பது எப்படி ஃபெட்ச்மெயிலுக்கு வந்தது என்பது பற்றிய ஒரு பொதுவான பாடம் இக்கதையில் உள்ளது. வழுத்திருத்தம் மட்டும்தான் இணையாக செய்யக்கூடியது (parallelizable) என்று இல்லை, மேம்பாடு மற்றும் (ஒருவேளை ஆச்சரியப்படும் அளவிற்கு) வடிவமைப்பு வகைகளைப் புத்தாய்வு செய்வதும் கூட. உங்கள் மேம்பாடு பாங்கு விரைவான மறுசெய்கை (rapidly iterative)

ஆகும்போது, நிரலாக்கம் மற்றும் மேம்பாடு வழுத்திருத்தத்தின் சிறப்பு நிகழ்வுகளாக மாறக்கூடும். அதாவது மென்பொருளின் அசல் திறன்கள் அல்லது கருத்தாக்கத்தில் ‘வழுக்களை’ சரிசெய்தல்.

வடிவமைப்பின் உயர் மட்டத்தில் இருந்தாலும், உங்கள் தயாரிப்புக்கு அருகில் உள்ள வடிவமைப்பு இடங்கள் வழியாக ஏராளமான இணை-நிரலாளர்கள் சீரற்ற முறையில் நடப்பது (random-walking through the design space) மிகவும் மதிப்புமிக்கதாக இருக்கும். ஒரு குட்டை நீர் எப்படி வடிகாலைக் கண்டுபிடிக்கிறது என்பதைக் கவனியுங்கள். அல்லது எறும்புகள் எப்படி உணவைக் கண்டுபிடிக்கின்றன என்பதைச் சிந்தித்துப் பாருங்கள். அடிப்படையில் பரவல் மூலம் ஆராய்தல் (exploration essentially by diffusion), அதைத் தொடர்ந்து பெரிதாக்கவல்ல தகவல் தொடர்பு உதவியுடன் வளங்களைப் பயன்படுத்துதல் (exploitation mediated by a scalable communication mechanism). ஹாரி ஹோச்ஹெய்சர் (Harry Hochheiser) மற்றும் என்னைப் போலவே இது நன்றாக வேலை செய்கிறது. நீங்கள் நுண்ணிய விவரங்களைப் பார்ப்பதற்காக மிக அருகில் இருந்தீர்கள் என்றால் வெளியிலுள்ளவர்களில் ஒருவர் ஒட்டுமொத்தமாகப் பார்த்து அருகிலேயே ஒரு பெரிய வெற்றியைக் காணக்கூடும்.



திறந்த மூல ஆவணங்களுக்குப் பங்களிப்பது எப்படி

ஃபெட்ச்மெயில் முதிர்ச்சி அடைகிறது

இப்படியாக ஃபெட்ச்மெயில் நேர்த்தியான மற்றும் புதுமையான வடிவமைப்புடன் மேம்பட்டு வந்தது. நான் தினமும் பயன்படுத்தியதால் எனக்குத் தெரிந்து நிரல் நன்றாக வேலை செய்தது. மேலும் பீட்டா பட்டியல் வளர்ந்து வந்தது. வேறு ஒரு சிலருக்கு மட்டுமே பயன்படக்கூடிய அற்பமான தனிப்பட்ட நிரல் திட்டத்தில் நான் ஈடுபடவில்லை என்பது படிப்படியாக எனக்குப் புரிந்தது. யூனிக்ஸ் கணினி மற்றும் SLIP/PPP அஞ்சல் இணைப்பு கொண்ட ஒவ்வொரு கொந்தருக்கும் உண்மையில் தேவைப்படும் ஒரு நிரலைத்தான் நான் கையில் எடுத்துள்ளேன்.

SMTP மேலனுப்புதல் அம்சத்துடன், அது போட்டி செயலிகளை மிகவும் முந்திச் சென்று “வகை ஆதிக்கவாதி (category killer)” ஆக முடியும். தன் பிரிவின் பணியை மிகவும் திறமையாக நிரப்பும் அந்த உன்னதமான நிரல்களில் ஒன்றாக ஆகிவிட்டது. மாற்றுகள் நிராகரிக்கப்படுவது மட்டுமல்ல, அனேகமாக மறந்தே போய்விடுகின்றன.

பிறரின் நல்ல யோசனைகளை அவர்களே எதிர்பார்த்ததற்கு மேல் எடுத்துச் செல்லும் பொறியியல் திறன்

இது போன்ற முடிவை உங்களால் உண்மையில்

குறிக்கோளாகவோ அல்லது திட்டமாகவோத் தொடங்க முடியாது என்று நினைக்கிறேன். மிகவும் சக்திவாய்ந்த வடிவமைப்பு யோசனைகளால் நீங்கள் அதில் இழுக்கப்பட வேண்டும். பின்னர் முடிவுகள் தவிர்க்க முடியாததாகவும், இயற்கையாகவும், முன்னரே தீர்மானிக்கப்பட்டதாகவும் தோன்றும். இதுபோன்ற யோசனைகளுக்கு முயற்சி செய்வதற்கான ஒரே வழி, நிறைய யோசனைகளைக் கொண்டிருப்பது. அல்லது பிறரின் நல்ல யோசனைகளை அவர்களே எதிர்பார்த்ததற்கு மேல் எடுத்துச் செல்லும் பொறியியல் திறனைக் கொண்டிருப்பதுதான்.

ஆண்டி டானென்பாம் (Andy Tanenbaum) IBM PC களுக்கேயான ஓர் எளிய யூனிக்ஸ் உருவாக்குவதற்கான முதலாவது யோசனையைக் கொண்டிருந்தார். இதை ஒரு கற்பித்தல் கருவியாகப் பயன்படுத்த உத்தேசித்தார். அவர் அதை மினிக்ஸ் (Minix) என்று அழைத்தார். லினஸ் டோர்வால்ட்ஸ் மினிக்ஸ் கருத்தை ஆண்டி ஒருகால் நினைத்ததை விட அதிகமாக முன்னேற்றினார். இதனால் அது அற்புதமான ஒன்றாக வளர்ந்தது. அதே வழியில் (சிறிய அளவில்தான் என்றாலும்), நான் கார்ல் ஹாரிஸ் மற்றும் ஹாரி ஹோச்ஹெய்சர் ஆகியோரின் சில யோசனைகளை எடுத்து அவற்றைத் தீவிரமாக முன்னேற்றினேன். நாங்கள் இருவரும் மக்கள் கற்பனை செய்வதுபோல 'அசல் மேதைகள்' அல்ல. கொந்தர் கட்டுக்கதைகளுக்கு மாறாக, பெரும்பாலான அறிவியல், பொறியியல் மற்றும் மென்பொருள் மேம்பாடு அசல் மேதைகளால் செய்யப்படுவதில்லை.

ஒவ்வொரு கொந்தரும் தங்கள் வாழ்நாளில் சாதிக்க முயலும் வெற்றியைப் போன்றே முடிவுகள் இருந்தன

உண்மையிலேயே ஒவ்வொரு கொந்தரும் தங்கள் வாழ்நாளில் சாதிக்க முயலும் வெற்றியைப் போன்றே முடிவுகள் கிளர்ச்சியூட்டுபவையாக இருந்தன! இதனால் எனது தரத்தை நான் இன்னும் உயர்த்த வேண்டியதாயிற்று. ஃபெட்ச்மெயிலை நான் இப்போது கண்டறிந்தது போல் சிறப்பாகச் செய்ய, எனது சொந்தத் தேவைகளுக்காக மட்டும் எழுதாமல், என்னை சுற்றியிருப்பவர்களைத் தாண்டியுள்ள மற்றவர்களுக்குத் தேவையான அம்சங்களையும் சேர்த்து ஆதரிக்க வேண்டும். அதே நேரத்தில் நிரலை எளிமையாகவும் வலுவாகவும் வைத்துக்கொண்டு அதைச் செய்யவேண்டும்.

இதை உணர்ந்த பிறகு நான் எழுதிய முதல் மற்றும் மிக முக்கியமான அம்சம் பலபெட்டியில் போடுதல் (multidrop). அதாவது, ஒரு பயனர் குழுவுக்கு அனைத்து அஞ்சல்களும் சேர்ந்திருக்கும் பெட்டிகளிலிருந்து அவற்றைத் திரளாகப் பெற்று, பின்னர் ஒவ்வொரு அஞ்சலையும் அவரவருக்கு அனுப்பும் திறன்.

பலபெட்டியில் போடுதல் ஆதரவைச் சேர்க்க நான் முடிவு செய்தேன். ஏனெனில் சில பயனர்கள் அதற்காகக் கூக்குரலிட்டனர். ஆனால் பெரும்பாலும் இது இரண்டுக்கும் பொதுவான நிரல் எழுதவேண்டி வருவதால் ஒருபெட்டியில் போடுதல் (single-drop) நிரலிலிருந்து வழுக்களை நீக்க வழிசெய்யும் என்று நினைத்தேன். அவ்வாறே நடந்தது. RFC 822 படி முகவரியை சரியாகப் பாகுபடுத்துவதற்கு (address parsing) எனக்கு மிக நீண்ட நேரம் எடுத்தது. அதில் எந்த ஒரு தனிப் பகுதியும் கடினமாக இருந்ததால் அல்ல, மாறாக அது ஒன்றுக்கொன்று சார்ந்த மற்றும் குழப்பமான விவரங்களின் குவியலைக் கொண்டிருந்ததால்.

உண்மையிலேயே ஒரு சிறந்த கருவி நீங்கள்

எதிர்பார்க்காத வகையிலும் பயன்தரவேண்டும்

ஆனால் பலபெட்டியில் போடுதல் ஒரு சிறந்த வடிவமைப்பு முடிவாகவும் ஆகியது. நான் எப்படி அறிந்தேன் என்பது இங்கே:

> மணிமொழி 14. எந்த ஒரு கருவியும் எதிர்பார்த்த விதத்தில் பயனுள்ளதாக இருக்க வேண்டும். ஆனால் உண்மையிலேயே ஒரு சிறந்த கருவி நீங்கள் எப்போதுமே எதிர்பார்க்காத வகையிலும் பயன்படவேண்டும்.

பலபெட்டியில் போடுதல் ஃபெட்ச்மெயிலின் எதிர்பாராத பயன் என்னவென்றால், இணைய இணைப்பின் பயனர் பக்கத்தில் வைத்திருக்கும் பட்டியலைக் கொண்டு அஞ்சல் பட்டியல்களை இயக்குதல் மற்றும் மாற்றுப்பெயர் விரிவாக்கம் (alias expansion) செய்தல். அதாவது ISP கணக்கின் மூலம் தனிநபர் கணினியை இயக்கும் ஒருவர், ISPயின் மாற்றுக் கோப்புகளை அணுகாமல் அஞ்சல் பட்டியலை நிர்வகிக்க முடியும்.

எனது பீட்டா-சோதனையாளர்கள் கோரும் மற்றொரு முக்கியமான மாற்றம் 8-பிட் MIME (8**it Multipurpose Internet Mail Extensions) செயல்பாட்டிற்கான ஆதரவாகும். இதைச் செய்வது மிகவும் எளிதானது. ஏனென்றால் நிரலை 8-பிட் சுத்தமாக வைத்திருப்பதில் நான் கவனமாக இருந்தேன் (அதாவது, ASCII எழுத்துக்குறி தொகுப்பில் பயன்படுத்தப்படாத 8வது பிட்டை நிரலுக்குள் தகவல்களை எடுத்துச் செல்லும் வேலைக்குப் பயன்படுத்தாமல்). இந்த அம்சத்திற்கான தேவையை நான் எதிர்பார்த்ததால் அல்ல, மாறாக மற்றொரு விதிக்குக் கீழ்ப்படிவதால்:

> மணிமொழி 15. எந்தவொரு

நுழைவாயில் மென்பொருளையும் எழுதும் போது, தரவுப் பாய்வை முடிந்தவரை மாற்றம் செய்யாமல் விட முயற்சி செய்யுங்கள். அதாவது பெறுபவர் உங்களைக் கட்டாயப்படுத்தினால் ஒழியத் தகவலைத் தூக்கி எறிய வேண்டாம்!

நான் இந்த விதிக்கு கீழ்ப்படியவில்லை என்றால், 8-பிட் MIME ஆதரவு கடினமாகவும் தரமற்றதாகவும் இருந்திருக்கும். ஆகவே, MIME தரநிலையைப் (RFC 1652) படித்து, தலைப்பு-உருவாக்குவது (header-generation) மட்டும்தான் நான் செய்ய வேண்டியிருந்தது.

சில ஐரோப்பிய பயனர்கள் ஓர் அமர்வுக்கு பெறப்படும் செய்திகளின் எண்ணிக்கையைக் கட்டுப்படுத்தும் விருப்பத்தைச் சேர்க்கச் சொல்லி என்னைத் தொல்லை செய்தனர் (இதன்மூலம் அவர்கள் தங்கள் விலையுயர்ந்த தொலைபேசிப் பிணையங்களில் செலவுகளைக் கட்டுப்படுத்த இயலும்). நான் இதை நீண்ட காலமாக எதிர்த்தேன், இன்னும் நான் அதைப் பற்றி முழுமையாகத் திருப்தியடையவில்லை. ஆனால் நீங்கள் உலகத்திற்காக எழுதுகிறீர்கள் என்றால், உங்கள் வாடிக்கையாளர்களுக்கு நீங்கள் செவிசாய்க்க வேண்டும்—அவர்கள் உங்கள் தயாரிப்பை விலை கொடுத்து வாங்குவதில்லை என்பதால் இது மாறாது.







ஒரு திறந்த மூல திட்டத்துக்கு பராமரிப்பாளராக ஆவது எப்படி

ஃபெட்ச்மெயில் கற்பித்த மேலும் சில பாடங்கள்

பொதுவான மென்பொருள்-பொறியியல் சிக்கல்களுக்குச் செல்வதற்கு முன், ஃபெட்ச்மெயில் அனுபவத்திலிருந்து இன்னும் சில குறிப்பிட்ட பாடங்கள் உள்ளன. தொழில்நுட்பத்தில் நாட்டமற்ற வாசகர்கள் இப்பகுதியைத் தவிர்க்கலாம்.

rc (கட்டுப்பாடு) கோப்பின் தொடரியல் (syntax) விருப்பமைவு செய்யக்கூடிய ‘இரைச்சல்’ குறிச்சொற்களை உள்ளடக்கியது. இவை பாகுபடுத்தியால் (parser) முற்றிலும் புறக்கணிக்கப்படும். இவை அனுமதிக்கும் ஆங்கிலம் போன்ற தொடரியல், பாரம்பரிய சுருக்கமான குறிச்சொல்-மதிப்பு இணைகளை (terse keyword-value pairs) விட, பெரும்பாலும் படிக்கக்கூடியதாக உள்ளது.

rc கோப்பு அறிவிப்புகள் எந்த அளவு ஒரு குறுமொழியை ஒத்திருக்கத் தொடங்குகின்றன என்பதை நான் கவனித்தபோது இவை இரவு நேரப் பரிசோதனையாகத் தொடங்கின. (இதனால்தான் பாப்கிளையன்டின் “வழங்கி (server)” என்ற குறிச்சொல்லை “கேள் (poll)” என மாற்றினேன்).

ஏவல் குறுமொழியை (imperative minilanguage) ஆங்கிலத்தைப் போலவே உருவாக்கலாமா?

அந்த ஏவல் குறுமொழியை ஆங்கிலத்தைப்

போலவே உருவாக்க முயற்சிப்பது அதை எளிதாகப் பயன்படுத்த வழி செய்யும் என்று எனக்குத் தோன்றியது. ஈமாக்ஸ் (Emacs), HTML மற்றும் பல தரவுத்தளப் பொறிகள் மூலம் எடுத்துக்காட்டும் வகையில், “அதை ஒரு மொழியாக்கு” வடிவமைப்பு கருத்தில் நான் உறுதியாக இருந்தாலும், நான் பொதுவாக “ஆங்கிலம் போன்ற” தொடரியல்களின் பெரிய ரசிகன் அல்ல.

பாரம்பரியமாக நிரலாளர்கள் மிகவும் துல்லியமான, கச்சிதமான மற்றும் மிகைமையே (redundancy) இல்லாத கட்டுப்பாட்டுத் தொடரியல்களுக்கு ஆதரவாக உள்ளனர். கணினி வளங்கள் விலையுயர்ந்த காலத்திலிருந்து இது ஒரு கலாச்சார மரபு. எனவே பாகுபடுத்தும் நிலைகள் (parsing stages) முடிந்தவரை மலிவானதாகவும் எளிமையாகவும் இருக்க வேண்டும். ஆங்கிலம், சுமார் 50% மிகைமையுடன், மிகவும் பொருத்தமற்ற மாதிரி போல இருந்தது.

கணினி மொழி கணினிக்கு எளிதாக இருப்பதை விட மனிதர்களுக்கு வசதியாக இருப்பது முக்கியம்

பொதுவாக ஆங்கிலம் போன்ற தொடரியல்களைத் தவிர்ப்பதற்கு இது எனது காரணம் அல்ல. அதைப் பொய்ப்பிப்பதற்காகவே இங்கு குறிப்பிடுகிறேன். கணினிகளின் கணிப்பி அளவு மற்றும் வேகம் மலிவாகி வருவதால், சுருக்கம் ஒரு குறிக்கோளாக இருக்கக்கூடாது. தற்காலத்தில் ஒரு கணினி மொழி கணினிக்கு எளிதாக இருப்பதை விட மனிதர்களுக்கு வசதியாக இருப்பது முக்கியம்.

இருப்பினும், எச்சரிக்கையாக இருக்க நல்ல காரணங்கள் உள்ளன. ஒன்று, பாகுபடுத்தும் கட்டத்தின் கடுஞ்சிக்கல்கள். இது வழுக்கள் மற்றும் பயனர் குழப்பத்தின் குறிப்பிடத்தக்க ஆதாரமாக இருக்கும் அளவுக்கு அதை உயர்த்த விரும்பவில்லை.

மற்றொன்று, ஒரு மொழியின் தொடரியலை ஆங்கிலம் போல உருவாக்க முயலும்போது, அது பேசும் “ஆங்கிலம்” வடிவம் இல்லாமல் கோணலாகியிருக்க வேண்டி வருகிறது. அதனால் இயற்கை மொழியின் மேலோட்டமான ஒற்றுமை பாரம்பரியத் தொடரியல் போலவே குழப்பமடைகிறது. (“நான்காம் தலைமுறை மொழிகள்” மற்றும் வணிகத் தரவுத்தள-வினவல் மொழிகள் (commercial database-query languages) எனப்படும் பலவற்றில் இந்த மோசமான விளைவை நீங்கள் காண்கிறீர்கள்.)

ஃபெட்ச்மெயில் கட்டுப்பாட்டுத் தொடரியல் இச்சிக்கல்களைத் தவிர்க்கிறது, ஏனெனில் மொழியின் களம் மிகவும் கட்டுப்படுத்தப்பட்டுள்ளது. இது ஒரு பொது-நோக்க மொழிக்கு அருகில் இல்லை. இது சொல்லும் சங்கதிகள் மிகவும் சிக்கலானவை அல்ல. எனவே ஆங்கிலத்தின் ஒரு சிறிய துணைப்பிரிவிற்கும் உண்மையான கட்டுப்பாட்டு மொழிக்கும் இடையில் மனதளவில் நகர்வதில் குழப்பம் ஏற்படுவதற்கான சாத்தியக்கூறுகள் குறைவு. இங்கே ஒரு பரந்த பாடம் இருக்கலாம் என்று நினைக்கிறேன்:

1. உங்கள் கணினி மொழி டுரிங் (Turing) முழுமைக்கு அருகில்கூட வராதபோது, தொடரியலை (syntax) எளிமையாக்குவதே சாலச்சிறந்தது.

தெளிவின்மை மூலம் பாதுகாப்பு (security by obscurity) கிடைக்குமா?

மற்றொரு பாடம் தெளிவின்மை மூலம் பாதுகாப்பு பற்றி. சில ஃபெட்ச்மெயில் பயனர்கள் rc கோப்பில் கடவுச்சொற்களை மறைகுறியீடாக்கி (encrypted) சேமிக்குமாறு மென்பொருளை மாற்றும்படி என்னிடம் கேட்டார்கள். ஏனென்றால் ஊடுருவிகள்

அவற்றை எளிதாகப் பார்க்க முடியாது.

நான் அதைச் செய்யவில்லை, ஏனெனில் இது உண்மையில் பாதுகாப்பைச் சேர்க்கவில்லை. உங்கள் rc கோப்பைப் படிப்பதற்கான அனுமதிகளைப் பெற்ற எவரும் எப்படியும் உங்களைப் போலவே ஃபெட்ச்மெயிலை இயக்க முடியும்—அவர்கள் உங்கள் கடவுச்சொல்லைப் பின்பற்றினால், அதைப் பெறுவதற்குத் தேவையான குறிவிலக்கியை (decoder) ஃபெட்ச்மெயில் நிரலிலிருந்தே அவர்களால் எடுக்க முடியும்.

.fetchmailrc கடவுச்சொல் மறைகுறியீடானது மிகவும் கடினமாக சிந்திக்காத நபர்களுக்கு இல்லாத பாதுகாப்பு உணர்வைக் கொடுக்கும். இங்கே பொதுவான விதி:

1. ஒரு பாதுகாப்பு அமைப்பு அதன் ரகசியம் அளவுதான் பாதுகாப்பானது. போலி ரகசியங்களில் எச்சரிக்கையாய் இருக்கவும்.

```
    }

    /**
-    * Returns a vlue from the session.
+    * Returns a value from the session.
     */
    public static String session(String key) {
        return session().get(key);
@@ -89,4 +89,4 @@ public static String flash(Strin
```

திறந்த மூலத்திற்கான எனது முதல் பங்களிப்பு மிகவும் சிறியது!

சந்தை பாணிக்கு அவசியமான முன்தேவைகள்

நிரலாளர்கள் ஓட்டிப்பார்த்து சோதிக்கக்கூடிய நிரல் தொடக்கத்திலேயே இருக்க வேண்டும்

இந்தக் கட்டுரைக்கான ஆரம்பகால மதிப்பாய்வாளர்கள் மற்றும் சோதனை பார்வையாளர்கள் வெற்றிகரமான சந்தை-பாணி வளர்ச்சிக்கான முன்தேவைகள் குறித்து தொடர்ந்து கேள்விகளை எழுப்பினர். இதில் திட்டத் தலைவரின் தகுதிகள் மற்றும் திட்டத்தை வெளியீடு செய்து இணை-நிரலாளர்கள் சமூகத்தை உருவாக்கத் தொடங்கும் போது உள்ள நிரலின் நிலை ஆகியவை அடங்கும்.

சந்தை பாணியில் அடிப்படையிலிருந்து ஒருவரால் நிரல் எழுத முடியாது என்பது மிகவும் தெளிவாக உள்ளது. சந்தை பாணியில் ஒருவர் சோதனை செய்யலாம், வழுத்திருத்தம் செய்யலாம் மற்றும் மேம்படுத்தலாம். ஆனால் சந்தை பயன்முறையில் ஒரு திட்டத்தை உருவாக்குவது மிகவும் கடினமாக இருக்கும். லினஸ் அதை முயற்சிக்கவில்லை. நானும் செய்யவில்லை. உங்கள் கூடிவரும் நிரலாளர் சமூகம் ஓட்டிப்பார்க்கக்கூடிய மற்றும் சோதிக்கக்கூடிய நிரல் தொடக்கத்தில் அவசியம் வேண்டும்.

நீங்கள் சமூகத்தை கட்டியெழுப்பத் தொடங்கும் போது, நீங்கள் முன்வைக்க வேண்டியது நம்பத்தகுந்த வாக்குறுதியாகும். உங்கள் திட்டம் சிறப்பாக செயல்பட வேண்டியதில்லை. இது கச்சாவாக, தரமற்ற, முழுமையற்ற மற்றும் சரியாக ஆவணப்படுத்தப்படாததாக இருக்கலாம். எதைச் செய்யத் தவறக்கூடாது என்றால் (அ) ஓடுவது, மற்றும் (ஆ) எதிர்காலத்தில் உண்மையிலேயே நேர்த்தியான ஒன்றாக உருவாகலாம் என்று சாத்தியமான இணை-நிரலாளர்களை நம்ப வைப்பது.

லினக்ஸ் மற்றும் ஃபெட்ச்மெயில் இரண்டும் வலுவான, கவர்ச்சிகரமான அடிப்படை வடிவமைப்புகளுடன் பொதுவில் வெளியிடப்பட்டன. நான் முன்வைத்த சந்தை மாதிரியைப் பற்றிச் சிந்திக்கும் பலர் இதை முக்கியமானதாகக் கருதினர். பின்னர் அதிலிருந்து உயர் மட்ட வடிவமைப்பு உள்ளுணர்வு மற்றும் திட்டத் தலைவரின் புத்திசாலித்தனம் இன்றியமையாதது என்ற முடிவுக்குத் தாவினர்.

ஆனால் லினஸ் தனது வடிவமைப்பை யூனிக்ஸ் இடம் இருந்து பெற்றார். நான் என்னுடையதை அதன் முன்னோடியான பாப்கிளையன்ட் இடமிருந்து பெற்றேன் (பின்னர் அது லினக்ஸை விடப் பெரிய அளவில் மாறியிருந்தாலும்). எனவே ஒரு சந்தை பாணி முயற்சிக்கான தலைவர்/ஒருங்கிணைப்பாளர் உண்மையில் விதிவிலக்கான வடிவமைப்புத் திறமையைக் கொண்டிருக்க வேண்டுமா அல்லது மற்றவர்களின் வடிவமைப்புத் திறமையைப் பயன்படுத்தியே அவரால் சமாளிக்க முடியுமா?

நல்ல வடிவமைப்புகளை உருவாக்குவதைவிட மற்றவர்களின் நல்ல யோசனைகளை அடையாளம் காண்பதுதான் முக்கியம்

ஒருங்கிணைப்பாளர் விதிவிலக்கான புத்திசாலித்தனத்துடன் வடிவமைப்புகளை உருவாக்க முடியும் என்பது முக்கியமானதல்ல என்று நான் நினைக்கிறேன், ஆனால் ஒருங்கிணைப்பாளர் மற்றவர்களிடமிருந்து நல்ல வடிவமைப்பு யோசனைகளை அடையாளம் காண முடியும் என்பது முற்றிலும் முக்கியமானது.

லினக்ஸ் மற்றும் ஃபெட்ச்மெயில் திட்டங்கள் இரண்டும் இதற்கான ஆதாரங்களைக் காட்டுகின்றன. லினஸ், ஓர் அற்புதமான அசல் வடிவமைப்பாளராக இல்லாவிட்டாலும், நல்ல வடிவமைப்பை அடையாளம் கண்டு அதை லினக்ஸ் கருநிரலில் ஒருங்கிணைக்க ஒரு சக்திவாய்ந்த திறமையைக் காட்டியுள்ளார். ஃபெட்ச்மெயிலில் (SMTP மேலனுப்புதல்) மிகவும் சக்திவாய்ந்த வடிவமைப்பு யோசனை வேறொருவரிடமிருந்து எப்படி வந்தது என்பதை நான் ஏற்கனவே விவரித்துள்ளேன்.

இந்தக் கட்டுரையின் ஆரம்பகாலப் பார்வையாளர்கள், சந்தை திட்டங்களில் வடிவமைப்பு அசல் தன்மையைக் குறைத்து மதிப்பிடுவதற்கான வாய்ப்புகள் என்னிடம் உள்ளன என்று கூறினர். ஏனெனில் என்னிடம் அத்திறமை நிறைய இருக்கிறது, எனவே நான் அதை ஒரு பொருட்டாக எடுத்துக் கொள்வதில்லை என்று என்னைப் பாராட்டினர். இதில் ஓரளவு உண்மை இருக்கலாம். வடிவமைப்பு (நிரல் அல்லது வழுத்திருத்தத்திற்கு மாறாக) நிச்சயமாக எனது வலிமையான திறமையாகும்.

ஆனால் மென்பொருள் வடிவமைப்பில் புத்திசாலித்தனமாகவும் அசலாகவும் இருப்பதில் உள்ள சிக்கல் என்னவென்றால், அது ஒரு பழக்கமாக ஆகிவிடுகிறது. நீங்கள் அவற்றை

வலுவாகவும் எளிமையாகவும் வைத்திருக்க வேண்டியிருக்கும் போது, நீங்கள் வடிவமைப்பை அழகாகவும் சிக்கலாகவும் மாற்றத் தொடங்குவீர்கள். நான் முன்னர் இத்தவறைச் செய்ததால் திட்டப்பணிகள் செயலிழந்துள்ளன. ஆனால் ஃபெட்ச்மெயில் திட்டத்தில் என்னால் இதைத் தவிர்க்க முடிந்தது.

எனவே நான் புத்திசாலித்தனமாக இருக்கும் என் போக்கை கட்டுப்படுத்தியதால், ஃபெட்ச்மெயில் திட்டம் ஓரளவு வெற்றியடைந்தது என்று நம்புகிறேன். வெற்றிகரமான சந்தைத் திட்டங்களுக்கு வடிவமைப்பு அசல் தன்மை அவசியம் என்பதற்கு எதிராக (குறைந்தபட்சம்) இது வாதிடுகிறது. மற்றும் லினக்ஸைக் கவனியுங்கள். லினஸ் டோர்வால்ட்ஸ் வளர்ச்சியின் போது இயங்குதள வடிவமைப்பில் அடிப்படைக் கண்டுபிடிப்புகளை இழுக்க முயன்றார் என்று வைத்துக்கொள்வோம். இதன் விளைவாக வரும் கருநிரல் நம்மிடம் இருப்பதைப் போலவே நிலையானதாகவும் வெற்றிகரமாகவும் இருக்கும் என்று தோன்றுகிறதா?

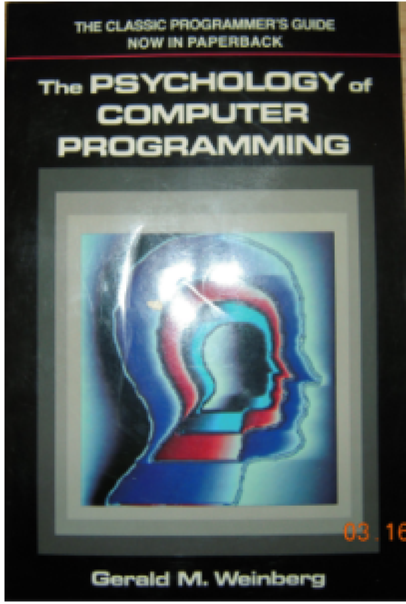
நிச்சயமாக, ஒரு குறிப்பிட்ட அடிப்படை அளவிலான வடிவமைப்பு மற்றும் நிரல் திறன் தேவை. ஆனால் சந்தை முயற்சியைத் தொடங்க வேண்டும் என்று தீவிரமாக நினைக்கும் எவரும் ஏற்கனவே அந்தக் குறைந்தபட்சத்துக்கு மேல் இருப்பார்கள் என்று நான் எதிர்பார்க்கிறேன். திறந்த மூல சமூகத்தின் நற்பெயரின் உள் சந்தையானது, பின்பற்றத் தகுதியற்ற வளர்ச்சி முயற்சிகளைத் தொடங்க வேண்டாம் என்று அவர்கள் மீது நுட்பமான அழுத்தம் கொடுக்கிறது. இதுவரை இது நன்றாக வேலை செய்துவருவதாகத் தெரிகிறது.

சந்தை திட்டத் தலைவருக்கு நல்ல மானிட உறவு மற்றும் தகவல் தொடர்பு திறன்கள் தேவை

சந்தைத் திட்டங்களுக்கு வடிவமைப்பு புத்திசாலித்தனம் போலவே மென்பொருள் மேம்பாட்டுடன் பொதுவாக தொடர்புபடுத்தப்படாத மற்றொரு வகையான திறன் முக்கியமானது என்று நான் நினைக்கிறேன். இது அதைவிட மிகவும் முக்கியமானதாக இருக்கலாம். ஒரு சந்தை திட்ட ஒருங்கிணைப்பாளர் அல்லது தலைவர் நல்ல மானிட உறவுத் திறன்கள் மற்றும் தகவல் தொடர்பு திறன்களைக் கொண்டிருக்க வேண்டும்.

இது வெளிப்படையாக இருக்க வேண்டும். ஒரு மேம்பாட்டு சமூகத்தை உருவாக்க, நீங்கள் மக்களை ஈர்க்க வேண்டும். நீங்கள் என்ன செய்கிறீர்கள் என்பதில் அவர்களுக்கு ஆர்வம் ஏற்பட வேண்டும். மேலும் அவர்கள் செய்யும் வேலையைப் பற்றி அவர்களை மகிழ்ச்சியாக வைத்திருக்க வேண்டும். தொழில்நுட்பத்தின் ஈர்ப்பு இதை நிறைவேற்றுவதற்கு நீண்ட தூரம் செல்லும். ஆனால் இது மட்டுமே ஒருக்காலும் போதாது. நீங்கள் வெளிப்படுத்தும் ஆளுமையும் முக்கியமானது.

லினஸ் ஒரு நயமான ஆளுமை. அவருடன் தொடர்பு கொள்ளவும், அவருக்கு உதவவும் விரும்புவது தற்செயல் நிகழ்வு அல்ல. நான் ஓர் உற்சாகமான புறநோக்கர் (extrovert). ஒரு கூட்டத்தில் பேசி அசத்துவதை ரசிக்கிறேன் மற்றும் ஒரு நகைச்சுவைப் பேச்சாளர் போன்ற உள்ளுணர்வுகளைக் கொண்டிருப்பது தற்செயல் நிகழ்வு அல்ல. சந்தை பாணி வேலை செய்ய, மக்களைக் கவர்வதில் குறைந்தபட்சம் கொஞ்சம் திறமை இருந்தால் அது பெரிதும் உதவுகிறது.



கணினி நிரலாக்கத்தின் உளவியல் – ஜெரால்ட் வெயின்பெர்க்

திறந்த மூல மென்பொருளின் சமூக சூழல்

ஒரு முக்கியமான பிரச்சினைக்குத் தீர்வு காண வேண்டுமென்றால் உங்களுக்கு சுவாரசியமான ஒரு பிரச்சினையில் தொடங்கவும்

இது என்றுமே பொய்யாகாது: சிறந்த நிரல்கள் படைப்பாளரின் அன்றாடப் பிரச்சினைகளுக்குத் தனிப்பட்ட தீர்வுகளாகத் தொடங்குகின்றன. பின்னர் இந்தப் பிரச்சினை ஒரு பெரிய வகுப்பினருக்குப் பொதுவானதாக இருப்பதால் பரவுகிறது. இது விதி 1 இன் கருத்துக்கு நம்மை மீண்டும் அழைத்துச் செல்கிறது, மிகவும் பயனுள்ள வகையில் இவ்வாறு மாற்றியும் கூறலாம்:

> மணிமொழி 18. ஒரு முக்கியமான பிரச்சினைக்குத் தீர்வு காண வேண்டுமானால், உங்களுக்கு சுவாரசியமாக இருக்கும் ஒரு சிக்கலைக் கண்டுபிடிப்பதன் மூலம் தொடங்கவும்.

அது கார்ல் ஹாரிஸ் மற்றும் முந்தைய பாப்கிளையண்டுக்கும் அப்படித்தான் இருந்தது. அதேபோல எனக்கும் ஃபெட்ச்மெயிலுக்கும் அப்படித்தான் இருந்தது. ஆனால் இது நீண்ட காலமாகப் புரிந்து கொள்ளப்பட்டதுதான்.

சுவாரசியமான விஷயம் என்னவென்றால், லினக்ஸ் மற்றும் ஃபெட்ச்மெயிலின் வரலாறுகளைப் பார்த்தால் நாம் இதன் அடுத்த கட்டத்தில் கவனம் செலுத்த வேண்டும். அதாவது பயனர்கள் மற்றும் இணை உருவாக்குநர்களின் பெரிய மற்றும் செயலில் உள்ள சமூகத்தின் முன்னிலையில் மென்பொருளின் பரிணாமம்.

"ஒரு நபரின் ஒரு மாத வேலை என்ற கட்டுக்கதை" நூலில் ஃபிரெட் புரூக்ஸ் நிரலாளர் நேரம் பரிமாற்றக் கூடியது அல்ல (not fungible) என்பதைக் கவனித்தார். ஆகவே தாமதமான மென்பொருள் திட்டத்தில் நிரலாளர்களைச் சேர்த்தால் அது மேலும் தாமதமாகும் என்றார். நாம் முன்பு பார்த்தது போல, ஒரு திட்டத்தின் சிக்கலான தன்மை மற்றும் தகவல் தொடர்பு செலவுகள் நிரலாளர்களின் எண்ணிக்கையின் வர்க்கத்துடன் உயர்கின்றன.(rise with the square of the number). அதே நேரத்தில் செய்யப்படும் வேலை நேரியல் ரீதியாக மட்டுமே உயரும் (rises linearly) என்று அவர் வாதிட்டார். புரூக்ஸின் சட்டம் உண்மைதான் என்று பரவலாகக் கருதப்படுகிறது. ஆனால் இக்கட்டுரையில் திறந்த மூல மேம்பாடு அதன் பின்னணியில் உள்ள அனுமானங்களைப் பொய்யாக்கும் பல வழிகளை நாம் ஆய்வு செய்துள்ளோம். மேலும், அனுபவ ரீதியாக, ப்ரூக்ஸின் சட்டம்தான் இந்தத் தலைப்பின் ஒட்டுமொத்தப் பார்வை என்றால் லினக்ஸே சாத்தியமில்லை.

நிரலாளர்கள் தங்கள் நிரலில் அதீத கட்டுப்பாடு **(over protective)** வைக்காத குழுக்களில் முன்னேற்றம் வியத்தகு முறையில் வேகமாக நடக்கிறது

ஜெரால்ட் வெயின்பெர்க்கின் (Gerald Weinberg) உன்னதமான கணினி நிரலாக்கத்தின் உளவியல் (The

Psychology of Computer Programming), பின்னறிவில் பார்த்தால், ப்ரூக்ஸுக்கு ஒரு முக்கிய திருத்தமாக நாம் பார்க்க முடியும். “தற்பெருமையற்ற நிரலாக்கம் (egoless programming)” பற்றிய அவரது விவாதத்தில், நிரலாளர்கள் தங்கள் நிரலைப் பற்றி மீயுடைமை உணர்வு (possessive) இல்லாத குழுக்களில், வழுக்கள் மற்றும் சாத்தியமான மேம்பாடுகளைத் தேடுவதற்கு மற்றவர்களை ஊக்குவித்தால், முன்னேற்றம் மற்ற இடங்களை விட வியத்தகு முறையில் வேகமாக நடக்கிறது என்று வெய்ன்பெர்க் கவனித்தார். (சமீபத்தில், கென்ட் பெக்கின் (Kent Beck) ‘அதீத நிரலாக்கம் (extreme programming)’ நுட்பம், நிரலாளர்கள் ஒருவருக்குப் பின்னால் நின்று ஒருவர் பார்த்துக் கொண்டு இணைகளாக வேலை செய்வது, இந்த விளைவைக் கட்டாயப்படுத்துவதற்கான முயற்சியாகக் கருதலாம்.)

வெய்ன்பெர்க் தேர்ந்தெடுத்த கலைச்சொற்கள் அவரது பகுப்பாய்விற்குத் தகுதியான அங்கீகாரம் பெறுவதற்குத் தடையாக இருந்திருக்கலாம். இணையக் கொந்தர்களை “தற்பெருமையற்றவர்கள் (’egoless)” என்று வர்ணிப்பதை நினைத்தால் சிரிப்புத்தான் வருகிறது. ஆனால் அவருடைய வாதம் முன்னெப்போதையும் விட இன்று மிகவும் அழுத்தமாகத் தெரிகிறது என்று நான் நினைக்கிறேன்.

சந்தை முறையானது, “தற்பெருமையற்ற நிரலாக்கம்” விளைவின் முழு ஆற்றலைப் பயன்படுத்துவதன் மூலம், ப்ரூக்ஸ் சட்டத்தின் விளைவை வலுவாகத் தணிக்கிறது. ப்ரூக்ஸ் சட்டத்தின் பின்னணியில் உள்ள கொள்கையை ரத்து செய்யவில்லை, ஆனால் ஒரு பெரிய நிரலாளர் எண்ணிக்கை மற்றும் மலிவான தகவல்தொடர்புகள் ஆகியவற்றைக் கொடுத்தால், அதன் விளைவுகள் மற்றபடி பார்க்க முடியாத போட்டியிடும் நேரியலற்ற

தன்மைகளால் (nonlinearities) மூழ்கடிக்கப்படலாம். இது நியூட்டன் மற்றும் ஐன்ஸ்டீன் இயற்பியலுக்கு இடையிலான உறவை ஒத்திருக்கிறது. பழைய அமைப்பு குறைந்த ஆற்றல்களில் இன்னும் செல்லுபடியாகும், ஆனால் நீங்கள் எடையையும் வேகத்தையும் போதுமான அளவு உயர்த்தினால், அணு வெடிப்புகள் அல்லது லினக்ஸ் போன்ற ஆச்சரியங்கள் கிடைக்கும்.

ஒரு மூடிய திட்டத்தில் தன் சொந்த மூளையை மட்டுமே பயன்படுத்தும் நிரலாளர் திறந்தமூலக் குழுக்களை விடப் பின்தங்கப் போகிறார்

லினக்ஸிலிருந்து நாம் என்ன கற்றுக்கொள்கிறோம் என்பதற்கு யூனிக்ஸின் வரலாறு நம்மைத் தயார்படுத்தியிருக்க வேண்டும் (மேலும் லினஸின் முறைகளை வேண்டுமென்றே காப்பியடிப்பதன் மூலம் நான் சோதனை ரீதியாக சிறிய அளவில் சரிபார்த்துள்ளேன்). அதாவது, நிரல் எழுதும் முறையானது ஒரு தனித்துச் செய்யும் செயலாக இருக்கும் அதே வேளையில், முழுச் சமூகத்தின் கவனத்தையும் மூளைத்திறனையும் பயன்படுத்துவதிலிருந்தே பெரும் மென்பொருட்கள் உருவாகின்றன. ஒரு மூடிய திட்டத்தில் தன் சொந்த மூளையை மட்டுமே பயன்படுத்தும் நிரலாளர், வடிவமைப்பு, நிரல் பங்களிப்புகள், வழு கண்டறிதல் மற்றும் பிற மேம்பாடுகளை ஆராயும் பின்னூட்டம் நூற்றுக் கணக்கான நிரலாளர்களிடமிருந்து (ஒருவேளை ஆயிரக் கணக்கானவர்கள்) வரும் திறந்த, பரிணாம சூழலை உருவாக்கத் தெரிந்த நிரலாளரை விடப் பின்தங்கப் போகிறார்.

ஆனால் பாரம்பரிய யூனிக்ஸ் உலகம் இந்த அணுகுமுறையை உச்ச நிலைக்கு உயர்த்துவதிலிருந்து பல காரணங்களால் தடுக்கப்பட்டது. ஒன்று பல்வேறு உரிமங்கள், வர்த்தக

ரகசியங்கள் மற்றும் வணிக நலன்களின் சட்டரீதியான தடைகள். இன்னொன்று (பின்னோக்கிப் பார்த்தால்) இணையம் அப்பொழுது இன்னும் முதிர்ச்சியடையவில்லை.

மலிவான இணையத்திற்கு முன், ஒரே இடத்திலிருந்த சில சமூகங்கள் இருந்தன. அங்கு கலாச்சாரம் வெயின்பெர்க்கின் “தற்பெருமையற்ற நிரலாக்கத்தை” ஊக்குவித்தது. மேலும் ஒரு நிரலாளர் பல திறமையான தலையீடு செய்பவர்கள் (kibitzers) மற்றும் இணை உருவாக்குநர்களை எளிதில் ஈர்க்க முடியும். பெல் லேப்ஸ் (Bell Labs), எம்ஐடி கணினி அறிவியல் மற்றும் செயற்கை நுண்ணறிவு ஆய்வகங்கள் (MIT LCS and AI labs), கலிஃபோர்னியா பல்கலை, பெர்க்லி (UC Berkeley). இவை பழம்பெரும் புதுப்புனைவுகளின் (innovations) இல்லமாக மாறியதுடன் இப்பொழுதும் சக்திவாய்ந்ததாக உள்ளன.

பரவலான இணைய அணுகலின் புதிய விதிகளின்படி எப்படி விளையாடுவது என்பதைக் கற்றுக்கொண்ட முதல் நபர் லினஸ் ஆவார்

முழு உலகையும் அதன் திறமைக் குழுவாகப் பயன்படுத்த நனவான மற்றும் வெற்றிகரமான முயற்சி எடுத்த முதல் திட்டம் லினக்ஸ் ஆகும். லினக்ஸின் கருவளர் காலம் உலகளாவிய வலையின் பிறப்புடன் ஒத்துப்போனது தற்செயல் நிகழ்வு என்று நான் நினைக்கவில்லை. மேலும் 1993-1994 ஆம் ஆண்டில் லினக்ஸ் அதன் தொடக்கநிலையை விட்டு வெளியேறிய அதே காலகட்டத்தில் ஐஎஸ்பி (ISP) தொழில்துறை நிறுவப்பட்டது மற்றும் இணையத்தில் பரவலான ஆர்வம் ஏற்பட்டது. பரவலான இணைய அணுகலை சாத்தியமாக்கிய புதிய விதிகளின்படி எப்படி விளையாடுவது என்பதைக் கற்றுக்கொண்ட முதல்

நபர் லினஸ் ஆவார்.

லினக்ஸ் மாதிரியின் வளர்ச்சிக்கு மலிவான இணையம் அவசியமான நிபந்தனையாக இருந்தபோதிலும், அது போதுமான நிபந்தனையாக இல்லை என்று நான் நினைக்கிறேன். மற்றொரு முக்கியமான காரணி, தலைமைத்துவ பாணி மற்றும் கூட்டுறவு பழக்கவழக்கங்களாகும். இது இணை-நிரலாளர்களை ஈர்க்கவும், ஊடகத்திலிருந்து அதிகபட்ச பயனைப் பெறவும் அனுமதிக்கும்.

ஆனால் இந்தத் தலைமைத்துவ பாணி என்ன, இந்தப் பழக்கவழக்கங்கள் என்ன? இவை அதிகார உறவுகளின் அடிப்படையில் இருக்க முடியாது. மேலும் இவை இருக்க முடிந்தாலும், வலுக்கட்டாயத் தலைமைத்துவம் நாம் பார்க்கும் முடிவுகளை உருவாக்காது. வெய்ன்பெர்க் 19 ஆம் நூற்றாண்டின் ரஷ்ய அரசின்மைவாதியான பியோட்டர் அலெக்சிவிச் க்ரோபோட்கினின் (Pyotr Alexeyvich Kropotkin) சுயசரிதையான “புரட்சியாளரின் நினைவலைகள் (Memoirs of a Revolutionist)” நூலை மேற்கோள் காட்டுகிறார்:

ஒரு நில அடிமைகளின் முதலாளி (serf-owner) குடும்பத்தில் வளர்ந்த நான், என் காலத்து இளைஞர்களைப் போலவே, பணியவைத்தல், கட்டளையிடுதல், திட்டுதல், தண்டித்தல் போன்றவற்றின் அவசியத்தில் மிகுந்த நம்பிக்கையுடன் வயதுவந்த வாழ்க்கையில் நுழைந்தேன். ஆனால் ஆரம்ப கட்டத்தில், நான் தீவிரமான நிறுவனங்களை நிர்வகிக்கவும், [கட்டற்ற] ஆண்களை சமாளிக்கவும் வேண்டியிருந்தது. மேலும் ஒவ்வொரு தவறும் ஒரே நேரத்தில் கடுமையான விளைவுகளுக்கு வழிவகுக்கும் போது, கட்டளை மற்றும் கட்டுப்படுத்தல் கொள்கையில் செயல்படுவது மற்றும் பொதுவான புரிதல்

கொள்கையில் செயல்படுவதற்கான வித்தியாசத்தை நான் பாராட்ட ஆரம்பித்தேன். முந்தையது ஒரு இராணுவ அணிவகுப்பில் வியக்கத்தக்க வகையில் செயல்படுகிறது. ஆனால் நிஜ வாழ்க்கையில் இதற்கு மதிப்பு கிடையாது. மேலும் பல ஒன்றிணைந்த மனத் திண்மைகளின் கடுமையான முயற்சியின் மூலம் மட்டுமே இலக்கை அடைய முடியும்.

லினக்ஸ் போன்ற ஒரு திட்டத்திற்குத் துல்லியமாகத் தேவைப்படுவது “பல ஒன்றிணைக்கும் மனத் திண்மைகளின் தீவிர முயற்சி (severe effort of many converging wills)”. இணையம் என்று நாம் அழைக்கும் அராஜகவாதிகளின் சொர்க்கத்தில் தன்னார்வலர்களிடையே “கட்டளைக் கொள்கையை” நடைமுறைப்படுத்த இயலாது. திறம்பட செயல்படவும், போட்டியிடவும், கூட்டுத் திட்டங்களை வழிநடத்தவும் விரும்பும் கொந்தர்கள், க்ரோபோட்கினின் “புரிந்துகொள்ளும் கொள்கை (principle of understanding)” மூலம் பரிந்துரைக்கப்பட்ட பயன்முறையில் ஆர்வமுள்ள பயனுள்ள சமூகங்களை எவ்வாறு கூட்டுவது மற்றும் உற்சாகப்படுத்துவது என்பதைக் கற்றுக் கொள்ள வேண்டும். அவர்கள் லினஸின் விதியைப் பயன்படுத்தக் கற்றுக்கொள்ள வேண்டும்.

லினஸின் விதிக்கான சாத்தியமான விளக்கமாக நான் முன்பு “டெல்பி விளைவு (Delphi effect)” என்று குறிப்பிட்டேன். ஆனால் உயிரியல் மற்றும் பொருளாதாரத்தில் தகவமைப்பு அமைப்புகளும் (adaptive systems) மிகவும் சக்திவாய்ந்த ஒப்புமைகளாகும். லினக்ஸ் உலகம் ஒரு தடையற்ற சந்தை (free market) அல்லது இயற்கை வாழும் நிலை (ecology) போன்று பல அம்சங்களில் செயல்படுகிறது. பயன்பாட்டை அதிகரிக்க முயற்சிக்கும் சுயநல முகவர்களின் ஒரு தொகுப்பு செயல்பாட்டில் தன்னைத்தானே திருத்திக் கொள்ளும் (self-correcting)

தன்னிச்சையான ஒழுங்கை (spontaneous order) உருவாக்குகிறது. இந்த ஒழுங்கு எவ்வளவுதான் மத்திய திட்டமிட்டாலும் அதைவிட அதிக விரிவானது மற்றும் திறமையானது. அப்படியிருக்க இங்கேதான் “புரிந்துகொள்ளும் கொள்கையைத்” தேட வேண்டும்.

கொந்தர்கள் அதிகப்படுத்துவது தற்பெருமை **(ego)** திருப்தி மற்றும் மற்ற கொந்தர்கள் மத்தியில் நற்பெயர்

லினக்ஸ் கொந்தர்கள் அதிகப்படுத்தும் “பயன்பாட்டு செயல்பாடு (utility function)” பாரம்பரியப் பொருளாதாரம் அல்ல. ஆனால் புலனாகாத அவர்களின் சொந்த தற்பெருமை (ego) திருப்தி மற்றும் மற்ற கொந்தர்கள் மத்தியில் நற்பெயருமாகும். (அவர்களது உந்துதலை “பொதுநல நோக்கு” என்று அழைக்கலாம், ஆனால் பொதுநல நோக்கு என்பது பரோபகாரருக்கு தற்பெருமை திருப்தியின் ஒரு வடிவம் என்ற உண்மையை இது புறக்கணிக்கிறது). இந்த வழியில் செயல்படும் தன்னார்வ கலாச்சாரங்கள் உண்மையில் வழக்கமில்லாதது அல்ல. நான் நீண்ட காலமாகப் பங்குபெரும் மற்றொன்று அறிவியல் புனைகதை ரசிகர் குழுவாகும் (science fiction fandom). இது கொந்தர்களைப் போலல்லாமல், தன்னார்வச் செயல்பாட்டின் அடிப்படை உந்துதலாக “தற்பெருமை-உயர்த்தலை” (ego**oosting – ஒருவரின் நற்பெயரை மற்ற ரசிகர்களுக்கிடையில் உயர்த்துதல்) நீண்டகாலமாக வெளிப்படையாக அங்கீகரித்துள்ளது.

லினஸ், தன்னை ஒரு திட்டத்தின் நுழைவாயில் காப்பாளராக வெற்றிகரமாக நிலைநிறுத்திக் கொண்டார். அதில் மேம்பாடு பெரும்பாலும் மற்றவர்களால் மேற்கொள்ளப்படுகிறது. மேலும்

அது தன்னிறைவு பெறும் வரை திட்டத்தில் ஆர்வத்தை வளர்த்தது அவர் க்ரோபோட்கினின் “பகிர்ந்து புரிந்துகொள்ளும் கொள்கையை (principle of shared understanding)” ஆழமாகப் புரிந்து கொண்டதைக் காட்டுகிறது. லினக்ஸ் உலகத்தைப் பற்றிய இந்தப் பொருளாதாரம் போன்ற (quasi-economic) பார்வை, அந்தப் புரிதல் எவ்வாறு பயன்படுத்தப்படுகிறது என்பதைப் பார்க்க நமக்கு உதவுகிறது.

லினஸின் முறை – தனிப்பட்ட கொந்தர்களின் சுயநலத்தை, நீடித்த ஒத்துழைப்பு தேவையான கடினமான நோக்கங்களுடன் இணைத்தல்

லினஸின் முறையானது “தற்பெருமை-உயர்த்தலில்” திறமையான சந்தையை உருவாக்குவதற்கான ஒரு வழியாக நாம் கருதலாம். அதாவது தனிப்பட்ட கொந்தர்களின் சுயநலத்தை, நீடித்த ஒத்துழைப்பால் மட்டுமே அடையக்கூடிய கடினமான நோக்கங்களுடன் முடிந்தவரை உறுதியாக இணைத்தல். அவரது முறைகளைக் காப்பியடித்து நல்ல முடிவுகளை அடையலாம் என்று ஃபெட்ச்மெயில் திட்டத்தில் நான் காட்டியுள்ளேன் (சிறிய அளவில் இருந்தாலும்). ஒரு வேளை நான் அவரை விட சற்று அதிக விழிப்புணர்வுடனும், முறையாகவும் கூடச் செய்திருக்கலாம்.

பலர் (குறிப்பாக கட்டற்ற சந்தைகளில் அரசியல் ரீதியாக அவநம்பிக்கை கொண்டவர்கள்) தன்னிச்சையாக இயங்கும் தற்பெருமையாளர்களின் கலாச்சாரம் துண்டு துண்டாக, பிராந்திய ரீதியாக, வீணாக்குவதாக, இரகசியமானதாக மற்றும் பகையுணர்வுடையதாக இருக்கும் என்று எதிர்பார்க்கிறார்கள். ஆனால் இந்த எதிர்பார்ப்பு லினக்ஸ் ஆவணப்படுத்தலின் ஆச்சரியமூட்டும் வகை, தரம் மற்றும் ஆழம் ஆகியவற்றால் (ஒரே ஓர்

எடுத்துக்காட்டு மட்டும்) தெளிவாகப் பொய்யாக்கப்பட்டுள்ளது. நிரலாளர்கள் ஆவணப்படுத்தலை வெறுக்கிறார்கள் என்பதை யாவரும் அறிவர். அப்படியானால், லினக்ஸ் கொந்தர்கள் எப்படி இவ்வளவு ஆவணங்களை உருவாக்குகிறார்கள்? வெளிப்படையாக லினக்ஸின் தற்பெருமை-உயர்த்தலில் உள்ள கட்டற்ற சந்தையானது, பெருமளவில் நிதிவசதியுடைய வணிக மென்பொருள் தயாரிப்பாளர்களின் ஆவணக் குழுக்களைக் காட்டிலும் நல்லொழுக்கமுள்ள, பரோபகார நடத்தைகளை உருவாக்க சிறப்பாகச் செயல்படுகிறது.

ஃபெட்ச்மெயில் மற்றும் லினக்ஸ் கருநிரல் திட்டங்கள் இரண்டும், பல கொந்தர்களின் தற்பெருமைகளுக்குச் சரியான வெகுமதி அளிப்பதன் மூலம், ஒரு பலமான நிரலாளர்/ ஒருங்கிணைப்பாளர் ஒரு திட்டம் பெருங்குழப்பத்தில் சிக்காமல் பல இணை-நிரலாளர்களைக் கொண்டிருப்பதன் பலன்களைப் பெற இணையத்தைப் பயன்படுத்தலாம் என்று காட்டுகின்றன. எனவே ப்ரூக்ஸின் சட்டத்திற்கு மாற்றுக் கருத்துகளாக நான் பின்வருவனவற்றை முன்மொழிகிறேன்:

> மணிமொழி 19: திட்டத்தின் ஒருங்கிணைப்பாளர் குறைந்தபட்சம் இணையம் போன்ற நல்ல தகவல்தொடர்பு ஊடகத்தைக் கொண்டிருந்து, மேலும் வற்புறுத்தலின்றி எப்படி வழிநடத்துவது என்பது தெரிந்திருந்தால், பல தலைவர்கள் இருப்பது ஒரே தலைவரை விடச் சிறந்தது.

மென்பொருளின் எதிர்காலம் பேராலயத்தை விட்டு வெளியேறிச் சந்தை பாணியைத் தழுவுபவர்களுக்குச் சொந்தமானதாக இருக்கும்

திறந்த மூல மென்பொருளின் எதிர்காலம் பெரும்பாலும் லினஸின் ஆட்டத்தை விளையாடத் தெரிந்தவர்களுக்கும், பேராலயத்தை விட்டு வெளியேறிச் சந்தை பாணியைத் தழுவுபவர்களுக்கும் சொந்தமானதாக இருக்கும் என்று நான் நினைக்கிறேன். தனிமனித தொலைநோக்கும் புத்திசாலித்தனமும் இனி முக்கியமில்லை என்று சொல்ல முடியாது. மாறாக, திறந்த மூல மென்பொருளின் அதிநவீன அம்சம் தனிப்பட்ட தொலைநோக்கு மற்றும் புத்திசாலித்தனத்திலிருந்து தொடங்கி, பின்னர் ஆர்வமுள்ள தன்னார்வ சமூகங்களை திறம்பட உருவாக்குவதன் மூலம் அதைப் பெருக்கும் நபர்களுக்குச் சொந்தமானது என்று நான் நினைக்கிறேன்.

ஒருவேளை இது திறந்த மூல மென்பொருளின் எதிர்காலம் மட்டுமல்ல. எந்தவொரு மூடிய மூல நிரல் குழுவும் லினக்ஸ் சமூகம் ஒரு பிரச்சினைக்குத் தீர்வுகாணக் கொண்டுவரும் திறமைசாலிகளின் தொகுப்பை ஒப்பிட முடியாது. ஃபெட்ச்மெயிலுக்குப் பங்களித்த 200 க்கும் மேற்பட்ட (1999: 600, 2000: 800) நபர்களை வேலைக்கு அமர்த்தக் கூட மிகச் சிலரால் மட்டுமே முடியும்!

ஒருவேளை இறுதியில் திறந்த மூல கலாச்சாரம் வெற்றிபெறும் காரணம் ஒத்துழைப்பு தார்மீக ரீதியாகச் சரியானது என்பதாலோ அல்லது மென்பொருள் “பதுக்கல்” தார்மீக ரீதியாகத் தவறானது என்பதாலோ அல்ல (பிந்தையதை நீங்கள் நம்பினாலும் லினஸும் நானும் நம்பவில்லை). திறந்த மூல சமூகம் ஒரு

பிரச்சினையில் திறமையான நிரலாளர்களின் நேரத்தைப் பல மடங்கு உள்ளிட முடியும். ஆகவே மூடிய மூல உலகம் திறந்த மூல சமூகங்களுடன் பரிணாம ஆயுதப் பந்தயத்தில் வெற்றிபெற முடியாது.


GNU Emacs
An extensible, customizable,
free/libre text editor — and more.
At its core is an interpreter for Emacs Lisp, a dialect of the Lisp programming language with extensions to support text editing.
↓ GNU/Linux
↓ Windows
↓ macOS

குனு ஈமாக்ஸ் தொகுப்பி

மேலாண்மையும் மேகினாட் கோடும்

1997 ஆம் ஆண்டின் முதலாவதான பேராலயமும் சந்தையும் ஆய்வுக்கட்டுரை முன்னர் உள்ள பார்வையுடன் முடிவடைந்தது. அதாவது இணையம் மூலம் தொடர்புகொள்ளும் நிரலாளர்/ புரட்சியாளர்களின் மகிழ்ச்சியான குழுக்கள் வழக்கமான மூடிய மென்பொருளின் கட்டளைமுறை உலகத்தைப் போட்டியில் வெல்லும் மற்றும் மூழ்கடிக்கும்.

பல அவநம்பிக்கை இயல்பு உள்ளவர்கள் இதை ஏற்கவில்லை. மேலும் அவர்கள் எழுப்பும் கேள்விகள் நியாயமான முறையில் ஈடுபடத் தகுதியானவை. சந்தை வாதத்திற்கான பெரும்பாலான ஆட்சேபனைகள், அதன் ஆதரவாளர்கள் வழக்கமான நிர்வாகத்தின் உற்பத்தித்திறன்-பெருக்க விளைவைக் (productivity-multiplying effect) குறைத்து மதிப்பிட்டுள்ளனர் என்ற கூற்றுக்கு வந்தது.

பாரம்பரிய எண்ணம் கொண்ட மென்பொருள்-மேம்பாடு மேலாளர்கள், திறந்த மூல உலகில் திட்டக் குழுக்கள் தற்போக்காக உருவாவது, மாறுவது மற்றும் கலைந்து போவது இயல்புநிலையாக இருக்கிறது என்று இதை எதிர்க்கிறார்கள். ஆகவே திறந்த மூல சமூகத்தில் நிரலாளர் எண்களின் அனுகூலம் இருப்பதுபோல் தோன்றினாலும் இப்பிரச்சினை

அதன் கணிசமான பாகத்தை இல்லாததாக்குகிறது. மென்பொருள் மேம்பாட்டில் உண்மையில் நெடுங்கால நீடித்த முயற்சி மற்றும் தயாரிப்புக்கான தொடர்ச்சியான முதலீட்டை எந்த அளவிற்கு எதிர்பார்க்கலாம் என்பதைத்தான் வாடிக்கையாளர்கள் கவனிப்பார்கள். லேசாகக் கொதித்துக் கொண்டிருக்கும் வடிசாறு சட்டியில் எவ்வளவு பேர் ஓர் எலும்பைப் போட்டுவிட்டுப் போனார்கள் என்பது ஒரு பொருட்டேயல்ல என்று சொல்கிறார்கள்.

இந்த வாதத்தில் ஏதோ நியாயம் இருப்பதாகத் தெரிகிறது. உண்மையில், மாயக் கொப்பரை (The Magic Cauldron) என்ற இந்த நூலின் மூன்றாம் பாகத்தில், மென்பொருள் தயாரிப்பின் பொருளாதாரத்திற்கு, எதிர்பார்க்கப்படும் எதிர்கால சேவை மதிப்புதான் முக்கியம் என்ற கருத்தை நான் உருவாக்கியுள்ளேன்.

குனு ஈமாக்ஸ் தொகுப்பி – நூற்றுக்கணக்கான பங்களிப்பாளர்கள் – **15** ஆண்டுகளுக்கும் மேலாக ஒருங்கிணைந்த கட்டுமானத் தொலைநோக்கு

ஆனால் இந்த வாதத்தில் ஒரு முக்கிய மறைமுகப் பிரச்சினையும் உள்ளது. திறந்த மூல மேம்பாடு அத்தகைய நீடித்த முயற்சியை வழங்க முடியாது என்பது அதன் மறைமுகமான அனுமானம். உண்மையில், வழக்கமான நிர்வாகம் இன்றியமையாததாகக் கருதும் வகையான ஊக்கக் கட்டமைப்புகள் (incentive structures) அல்லது நிறுவனக் கட்டுப்பாடுகள் (institutional controls) இல்லாமல் ஒரு ஒத்திசைவான திசையையும், பயனுள்ள பராமரிப்பாளர் சமூகத்தையும் நீண்ட காலத்திற்குப் பராமரித்த திறந்த மூல திட்டங்கள் உள்ளன. குனு ஈமாக்ஸ் (GNU Emacs) நிரல் தொகுப்பியின் வளர்ச்சி ஒரு தீவிரமான மற்றும்

கற்றுக்கொள்ளக்கூடிய எடுத்துக்காட்டு. நூற்றுக்கணக்கான பங்களிப்பாளர்களின் முயற்சிகளை 15 ஆண்டுகளுக்கும் மேலாக ஓர் ஒருங்கிணைந்த கட்டுமானத் தொலைநோக்கில் (unified architectural vision) உள்வாங்கியது. அதிகம் பேர் விட்டுச்சென்ற போதிலும், ஒரே ஒரு நபர் (அதன் படைப்பாளர்) மட்டுமே அந்தக் காலம் முழுதும் தொடர்ந்து செயலில் இருந்தார். எந்த மூடிய மூலத் தொகுப்பியும் இந்த நீண்ட ஆயுட்கால சாதனையை நெருங்கவில்லை.

பேராலயமா சந்தையா என்ற பயன்முறை தொடர்பான மற்ற வாதங்கள் தவிர வழக்கமான முறையில் நிர்வகிக்கப்படும் மென்பொருள் மேம்பாட்டின் நன்மைகளைக் கேள்விக்குட்படுத்த இது ஒரு தனி காரணம். குனு ஈமாக்ஸ் (GNU Emacs) 15 ஆண்டுகளுக்கும் மேலாக நிலையான கட்டுமானத் தொலைநோக்கை வெளிப்படுத்துகிறது மற்றும் லினக்ஸ் போன்ற இயங்குதளம் 8 ஆண்டுகளுக்கும் மேலாக வேகமாக மாறிவரும் வன்பொருட்களுக்கு ஆதரவு அளித்து இயங்குதள தொழில்நுட்பத்தை வெளிப்படுத்துகிறது. 5 ஆண்டுகளுக்கும் மேலாக பல நன்கு கட்டமைக்கப்பட்ட திறந்த மூல திட்டங்கள் இருக்கின்றன. இவை சாத்தியம் என்றால் வழக்கமாக நிர்வகிக்கப்படும் வளர்ச்சியின் மிகப்பெரிய மேற்செலவு உண்மையில் எதைத்தான் வாங்குகிறது என்று கேட்பதற்கு நமக்கு உரிமை உள்ளது.

காலக்கெடு, செலவுத் திட்டம், அம்சப் பட்டியல் ஆகிய மூன்று இலக்குகளில் ஒன்றைக் கூட பூர்த்தி செய்வது ‘நிர்வகிக்கப்பட்ட’ திட்டத்தில் அரிது

அது எதுவாக இருந்தாலும், காலக்கெடு, செலவுத் திட்டம், அம்சப் பட்டியல் ஆகிய மூன்று இலக்குகளை நம்பத்தகுந்த முறையில் செயல்படுத்துவது இருக்கட்டும், ஆனால் இவற்றில் ஒன்றைக் கூட

பூர்த்தி செய்வது ஓர் அரிய ‘நிர்வகிக்கப்பட்ட’ திட்டமாகும். திட்டத்தை செயல்படுத்தும் காலத்தின்போது தொழில்நுட்பம் மற்றும் பொருளாதார சூழலில் ஏற்படும் மாற்றங்களுக்கு ஏற்ப இது செயல்படும் திறன் கொண்டதாகத் தெரியவில்லை. திறந்த மூல சமூகம் அந்த மதிப்பெண்ணில் மிகவும் பயனுள்ளதாக நிரூபிக்கப்பட்டுள்ளது (எடுத்துக்காட்டாக, இணையத்தின் 30 ஆண்டுகால வரலாற்றை தனியுரிம பிணையத் தொழில்நுட்பங்களின் குறுகிய ஆயுளுடன் ஒப்பிடுவதன் மூலம் ஒருவர் உடனடியாகச் சரிபார்க்க முடியும். மைக்ரோசாஃப்ட் விண்டோஸில் 16-பிட் இலிருந்து 32-பிட் மாற்றத்துக்கு ஆன செலவையும், அதே காலகட்டத்தில் இன்டெல் மட்டுமல்லாமல் 64-பிட் ஆல்பா உட்பட பன்னிரண்டுக்கும் மேற்பட்ட வன்பொருட்களுக்கு லினக்ஸ் பெரிதும் சிரமமின்றி மேல்நோக்கி இடம்பெயர்ந்ததையும் (upward migration) ஒப்பிட்டுப்பார்க்கலாம்).

யார் மீதாவது வழக்குத் தொடர்வதன் மூலம் ஆறுதல் அடைவது உங்கள் குறிக்கோள் அல்ல – உங்களுக்கு வேலை செய்யும் மென்பொருள் வேண்டும்

பாரம்பரிய முறையில் தாங்கள் வாங்குவதாகப் பலர் நினைக்கும் ஒரு சங்கதி, யாரோ ஒருவர் சட்டப்பூர்வமாக பொறுப்பேற்க வேண்டும் மற்றும் திட்டத்தில் தவறு நடந்தால் இழப்பீடு பெறலாம் என்பது. ஆனால் இது ஒரு மாயை. பெரும்பாலான மென்பொருள் உரிமங்கள், செயல்திறன் ஒருபுறம் இருக்கட்டும், வணிகத்திறனுக்கான உத்தரவாதத்தைக் (warranty of merchantability) கூட மறுப்பதற்காக எழுதப்பட்டவை. மேலும் மென்பொருள் சரியாகச் செயல்படவில்லை என்று வெற்றிகரமாக இழப்பீடு பெறுதல் மிக மிக அரிது.

அது பொதுவானதாகவே இருந்தாலும், யார் மீதாவது வழக்குத் தொடர்வதன் மூலம் ஆறுதல் அடைவது உங்கள் குறிக்கோள் அல்ல. உங்கள் விருப்பம் ஒரு வழக்கில் மாட்டிக்கொள்வது அல்ல. உங்களுக்கு வேலை செய்யும் மென்பொருள் வேண்டும்.

அப்படியென்றால் அந்த நிர்வாகத்திற்காக மேற்செலவு செய்வதன் மூலம் எதைத்தான் வாங்குகிறோம்?

அதைப் புரிந்துகொள்வதற்கு, மென்பொருள் மேம்பாட்டு மேலாளர்கள் என்ன நம்புகிறார்கள் என்பதை நாம் புரிந்து கொள்ள வேண்டும். எனக்குத் தெரிந்த, இந்த மென்பொருள் மேம்பாட்டு மேலாளர் வேலையில் மிகவும் நன்றாக செயல்படுவதாகத் தெரியும் ஒரு பெண்மணி மென்பொருள் திட்ட மேலாண்மைக்கு ஐந்து செயல்பாடுகள் உள்ளன என்று சொல்கிறார்கள்:

- இலக்குகளை வரையறுத்து, அனைவரையும் ஒரே திசையில் முயற்சி செய்ய வைத்தல்.
- திட்டத்தைக் கண்காணித்து முக்கியமான விவரங்கள் விட்டுப்போகாமல் பார்த்துக் கொள்ளுதல்.
- சலிப்பூட்டும் ஆனால் அவசியமான கசப்பான வேலைகளைச் செய்ய ஊழியர்களை ஊக்குவித்தல்.
- சிறந்த உற்பத்தித்திறனுக்காக ஊழியர்களைப் பயன்படுத்த ஏற்பாடு செய்தல்.
- திட்டத்தை நிலைநிறுத்தத் தேவையான வளங்களைத் திரட்டுதல்.

இவை அனைத்தும் வெளிப்படையாகத் தகுதியான இலக்குகள்தான். ஆனால் திறந்த மூல மாதிரியில்

மற்றும் அதைச் சுற்றியுள்ள சமூக சூழலில், இவை விசித்திரமான பொருத்தமற்றதாகத் தோன்றலாம். இவற்றைப் பின்னிருந்து முன் வரிசையில் பார்ப்போம்.

வளங்களைத் திரட்டுதல் (resource marshalling) பெரும்பாலும் அடிப்படையில் தற்காப்பு என்று என் நண்பர் சொல்கிறார். உங்கள் ஊழியர்கள், இயந்திரங்கள் மற்றும் அலுவலக இடம் உங்களிடம் இருந்தால், அதே வளங்களுக்காகப் போட்டியிடும் சக மேலாளர்களிடமிருந்தும், இருக்கும் அளவான வளங்களை மிகவும் திறமையாகப் பயன்படுத்த ஒதுக்க முயற்சிக்கும் உயர் அதிகாரிகளிடமிருந்தும் நீங்கள் அவற்றைப் பாதுகாக்க வேண்டும்.

ஆனால் திறந்த மூல நிரலாளர்கள் தன்னார்வத் தொண்டர்கள். தாங்கள் பணிபுரியும் திட்டங்களுக்கு பங்களிக்கும் ஆர்வம் மற்றும் திறன் ஆகிய இரண்டும் அவர்களே தேர்ந்தெடுத்தது (அவர்கள் திறந்த மூலத்தில் வேலை செய்யச் சம்பளம் பெறும்போது கூட இது பொதுவாக உண்மையாக இருக்கும்.) தன்னார்வ நெறிமுறைகள் தானாக வளங்களைத் திரட்டுதலின் ‘தாக்குதல்’ பக்கத்தைக் கவனித்துக் கொள்கின்றன. அவர்கள் தங்கள் சொந்த வளங்களைத் திட்டத்துக்குக் கொண்டு வருகிறார்கள். ஒரு மேலாளர் வழக்கமான பொருளில் ‘தற்காப்பு விளையாடுதல் (play defense)’ சிறிதளவு கூடத் தேவையில்லை.

எப்படியிருந்தாலும், மலிவான தனிநபர் கணினிகள் மற்றும் வேகமான இணைய இணைப்புகளின் உலகில், திறமையான கவனம் மட்டுமே உண்மையில் வரம்புக்குட்படுத்தும் ஒரே வளம் என்பதை நாம் தொடர்ந்து காண்கிறோம். கணினிகள், இணைய இணைப்புகள் அல்லது அலுவலக இடம் தேவைக்காக ஒருபோதும் திறந்த

மூலத் திட்டங்கள் தடுமாறுவதில்லை. நிரலாளர்கள் ஆர்வத்தை இழக்கும்போது மட்டுமே அவை நின்று போகின்றன.

திறந்த மூலம் வெற்றிகரமாக இருக்க ஒரு காரணம் அதன் கலாச்சாரம் மிகவும் திறமையான **5%** நிரலாளர்களை மட்டுமே ஏற்றுக்கொள்வது

அப்படி இருக்கையில், திறந்த மூலக் கொந்தர்கள் சுய-தேர்வு மூலம் அதிகபட்ச உற்பத்தித்திறனுக்காகத் தங்களை ஒழுங்கமைத்துக் கொள்வது இரட்டிப்பாக முக்கியமானது. மேலும் இதன் சமூக சூழல் தயவு தாட்சணியம் பார்க்காமல் திறமையைத் தேர்ந்தெடுக்கிறது. திறந்த மூலம் வெற்றிகரமாக இருப்பதற்கு ஒரு காரணம் அதன் கலாச்சாரம் மிகவும் திறமையான 5% நிரலாளர்களை மட்டுமே ஏற்றுக்கொள்வது என்று திறந்த மூல உலகம் மற்றும் பெரிய மூடிய திட்டங்கள் இரண்டையும் நன்கு அறிந்த எனது நண்பர் நம்புகிறார். அவர் தனது பெரும்பாலான நேரத்தை மற்ற 95% நிரலாளர்களைப் பயன்படுத்த ஏற்பாடு செய்வதில் செலவழிக்கிறார். மேலும் மிகவும் திறமையான நிரலாளர்கள் மற்றும் தகுதிவாய்ந்தவர்களுக்கிடையில் உற்பத்தித்திறனில் நன்கு அறியப்பட்ட நூறு மடங்கு வேறுபாட்டை நேரடியாகக் கண்டிருக்கிறார்.

அந்த வேறுபாட்டின் அளவு எப்போதுமே ஒரு சங்கடமான கேள்வியை எழுப்புகிறது. தனிப்பட்ட திட்டங்களும் ஒட்டுமொத்தத் துறையும் திறமையின் அடிமட்டத்தில் 50% க்கும் அதிகமானவர்கள் இல்லாமல் இருந்தால் சிறப்பாக இருக்குமா? வழக்கமான மென்பொருள் நிர்வாகத்தின் ஒரே செயல்பாடு நிகர இழப்பிலிருந்து குறைந்த வெற்றிக்கு மாற்றுவதுதான் என்றால் இந்த வேலைக்கு மதிப்பே கிடையாது என்பதை சிந்தனைமிக்க மேலாளர்கள் நீண்ட காலமாகப்

புரிந்துகொண்டுள்ளனர்.

வேறு ஏதாவது செய்ய விரும்புபவர்கள் நிறைந்த பல கட்டடங்களை நிர்வகிப்பதை விட, இணையத்தில் இருந்து சுயமாகத் தேர்ந்தெடுக்கப்பட்ட தன்னார்வலர்களை ஆட்சேர்ப்பு செய்வது மலிவானது மற்றும் பயனுள்ளது என்பதற்கான கடினமான ஆதாரங்களை வழங்குவதன் மூலம் திறந்த மூல சமூகத்தின் வெற்றி இந்தக் கேள்வியை கணிசமாகக் கூர்மையாக்குகிறது.

இது உந்துதல் பற்றிய கேள்விக்கு நம்மை நேர்த்தியாகக் கொண்டுவருகிறது. எனது நண்பரின் கருத்தைக் கூறுவதற்குச் சமமான மற்றும் அடிக்கடி கேட்கப்படும் வழி என்னவென்றால், பாரம்பரிய வளர்ச்சி மேலாண்மை என்பது மோசமான ஊக்கம் கொண்ட நிரலாளர்களுக்குத் தேவையான ஈடுசெய்தல் ஆகும். இல்லையேல் அவர்கள் நல்லபடி வேலையைச் செய்ய மாட்டார்கள்.

இந்த பதில் பொதுவாக ‘கவர்ச்சியான’ அல்லது தொழில்நுட்ப ரீதியாக இனிமையான வேலையைச் செய்ய மட்டுமே திறந்த மூல சமூகத்தை நம்பியிருக்க முடியும் என்ற கூற்றுடன் பயணிக்கிறது. பணத்தால் தூண்டப்பட்ட குறுவறை (cubicle) ஏவலர்களை மேலாளர்கள் விரட்டி வேலைவாங்கினால் தவிர, மற்ற வேலைகள் எதுவும் செய்யாமல் விடப்படும் (அல்லது மோசமாக மட்டுமே செய்யப்படும்). சிந்தனைக் கோளத்தில் நம் பகுதியைக் குறித்தல் (Homesteading the Noosphere) என்ற இந்நூலின் இரண்டாம் பாகத்தில் இந்தக் கூற்றில் சந்தேகம் இருப்பதற்கான உளவியல் மற்றும் சமூகக் காரணங்களை நான் குறிப்பிடுகிறேன். இருப்பினும், தற்போதைய நோக்கங்களுக்காக, அதை உண்மையாக ஏற்றுக்கொள்வதன் தாக்கங்களைச் சுட்டிக்காட்டுவது மிகவும் சுவாரசியமானது என்று

நினைக்கிறேன்.

வழக்கமான, மூடிய-மூல, பெரிய செலவில் நிர்வகிக்கப்படும் மென்பொருள் மேம்பாட்டின் பாணியானது சலிப்பூட்டும் வேலைகள் என்ற ஒருவித மேகினாட் கோட்டினால் (Maginot Line) மட்டுமே பாதுகாக்கப்படுகிறதா? அப்படியென்றால் ஒவ்வொரு தனிப்பட்ட பயன்பாட்டுப் பகுதியிலும் யாரும் இந்தச் சிக்கல்கள் மிகவும் சுவாரசியமானவை என்று கண்டுபிடிக்காத வரை மற்றும் வேறு யாரும் அவற்றைச் சுற்றி எந்த வழியையும் கண்டுபிடிக்காத வரைதான் அது சாத்தியமானதாக இருக்கும். ஒரு ‘சலிப்பூட்டும்’ மென்பொருளுக்கு திறந்த மூலப் போட்டி இருக்கும் தருணத்தில், அந்த சிக்கலைத் தீர்ப்பதற்காகவே அதைத் தேர்ந்தெடுத்த ஒருவரால் அது இறுதியாகக் கையாளப்பட்டது என்பதை வாடிக்கையாளர்கள் அறிந்து கொள்ளப் போகிறார்கள். இது, மென்பொருளில், மற்ற வகையான ஆக்கப்பூர்வமான வேலைகளைப் போல, பணத்தை மட்டுமே விட மிகவும் பயனுள்ள உந்துதலாக இருக்கிறது.

ஊக்குவிப்பதற்காக மட்டுமே ஒரு வழக்கமான மேலாண்மை கட்டமைப்பைக் கொண்டிருப்பது, ஒருவேளை நல்ல உத்தி, ஆனால் மோசமான வியூகம். ஒரு குறுகிய கால வெற்றி, ஆனால் நீண்ட காலத்தில் ஓர் உறுதியான இழப்பு.

இதுவரை, வழக்கமான மேம்பாடு மேலாண்மை என்பது திறந்த மூலத்திற்கு எதிராக இரண்டு பகுதிகளில் (வளங்களைத் திரட்டுதல், அமைத்தல்) ஒரு தோல்வியடையும் பந்தயம் போலவும், மூன்றில் ஒரு பகுதியில் (உந்துதல்) கடன் வாங்கிய நேரத்தைப் போலவும் இருக்கிறது. மேலும் மோசமான தொல்லைக்குள்ளான வழக்கமான மேலாளர் கண்காணிப்பு பிரச்சினையில் இருந்து எந்த

உதவியும் பெறப் போவதில்லை. திறந்த மூல சமூகத்தின் வலுவான வாதம் என்னவென்றால், பரவலாக்கப்பட்ட சக மதிப்பாய்வு விவரங்கள் நழுவாமல் இருப்பதை உறுதி செய்வதற்கான அனைத்து வழக்கமான முறைகளையும் வெல்கிறது.

வழக்கமான மென்பொருள் திட்ட நிர்வாகத்தின் மேல்செலவுக்கான நியாயமாக இலக்குகளை வரையறுப்பதை நாம் ஒதுக்க முடியுமா? ஒருவேளை; ஆனால் அவ்வாறு செய்ய, திறந்த மூல உலகில் ஒத்த பங்கை நிரப்பும் திட்டத் தலைவர்கள் மற்றும் மரபுப் பெரியவர்களைக் காட்டிலும் நிர்வாகக் குழுக்கள் மற்றும் நிறுமச் செயல் திட்டங்கள் தகுதியான மற்றும் பரவலாகப் பகிரப்பட்ட இலக்குகளை வரையறுப்பதில் மிகவும் வெற்றிகரமானவை என்று நம்புவதற்கு நமக்கு நல்ல காரணம் தேவை.

இது முதல் நோக்கில் மிகவும் கடினமான வழக்கு. மேலும் இதைக் கடினமாக்குவது தராசின் திறந்த மூலப் பக்கம் அல்ல (இமாக்ஸின் நீண்ட ஆயுட்காலம் அல்லது "உலக ஆதிக்கம்" என்ற பேச்சு மூலம் நிரலாளர்களின் கூட்டத்தைத் திரட்டும் லினஸ் டோர்வால்ட்ஸின் திறன்). மாறாக, இது மென்பொருள் திட்டங்களின் இலக்குகளை வரையறுப்பதற்கான வழக்கமான வழிமுறைகளின் நிரூபிக்கப்பட்ட பரிதாபம்.

வழக்கமான மென்பொருள் திட்டங்களில் **60%** முதல் **75%** வரை முடிக்கப்படுவதில்லை அல்லது அவற்றின் பயனர்களால் நிராகரிக்கப்படுகின்றன

மென்பொருள் பொறியியலின் மிகவும் பிரபலமான பொதுப்பேச்சு கோட்பாடுகளில் ஒன்று, வழக்கமான மென்பொருள் திட்டங்களில் 60% முதல் 75% வரை முடிக்கப்படுவதில்லை அல்லது அவற்றின் பயனர்களால் நிராகரிக்கப்படுகின்றன. அந்த வரம்பு எங்காவது உண்மையாக இருந்தால் (அதை மறுக்கும்

எந்தவொரு அனுபவத்தின் மேலாளரையும் நான் சந்தித்ததில்லை) பின்னர் பெரும்பாலான திட்டங்கள் (அ) யதார்த்தமாக அடைய முடியாத அல்லது (ஆ) தவறான இலக்குகளை இலக்காகக் கொண்டுள்ளன.

மற்ற பிரச்சினைகளை விட இதுவே, இன்றைய மென்பொருள் பொறியியல் உலகில் “மேலாண்மைக் குழு” என்ற வாக்கியம் கேட்பவரின் முதுகுத் தண்டுவடத்தில் குளிர்ச்சியை ஏற்படுத்தக்கூடியதாக உள்ளது, கேட்பவர் மேலாளராக இருந்தாலும் கூட (அல்லது குறிப்பாக). நிரலாளர்கள் மட்டுமே இந்த முறையைப் பற்றி குறை கூறிய நாட்கள் போய்விட்டன. தில்பர்ட் (Dilbert) கார்ட்டூன்கள் இப்போது நிர்வாகிகளின் மேசைகளின்மேல் தொங்குகின்றன.

அப்படியானால், பாரம்பரிய மென்பொருள் மேம்பாட்டு மேலாளருக்கு எங்கள் பதில் எளிமையானது. திறந்த மூல சமூகம் வழக்கமான நிர்வாகத்தின் மதிப்பை உண்மையில் குறைத்து மதிப்பிட்டிருந்தால், உங்களில் பலர் ஏன் உங்கள் சொந்த செயல்முறைக்கு அவமதிப்பைக் காட்டுகிறீர்கள்?

மீண்டும் ஒருமுறை திறந்த மூல சமூகத்தின் எடுத்துக்காட்டு இந்தக் கேள்வியைக் கணிசமாகக் கூர்மைப்படுத்துகிறது. ஏனெனில் நாங்கள் செய்வதை விளையாட்டாகச் செய்கிறோம். எங்களின் ஆக்கப்பூர்வமான வேலை தொழில்நுட்பம், சந்தைப்-பங்கு மற்றும் மனப்-பங்கு வெற்றிகளை அசுர வேகத்தில் குவித்து வருகிறது. எங்களால் சிறந்த மென்பொருளை உருவாக்க முடியும் என்பதை மட்டுமல்ல, மகிழ்ச்சி ஒரு சொத்து என்பதையும் சேர்த்து நிரூபித்து வருகிறோம்.

இந்தக் கட்டுரையின் முதல் பதிப்பிற்கு இரண்டரை ஆண்டுகளுக்குப் பிறகு, நான்

முடிப்பதற்குமுன் வைக்கக்கூடிய மிகத் தீவிரமான சிந்தனை, திறந்த மூலம் மென்பொருள் உலகில் ஆதிக்கம் செலுத்தும் என்பது அல்ல. எல்லாவற்றிற்கும் மேலாக, இக்காலத்தில் நிதானமான பெரு நிறுவன மேலதிகாரிகள் பலருக்கு இது நம்பத்தகுந்ததாகத் தெரிகிறது.

திறந்த மூல வெற்றியின் மிக முக்கியமான விளைவு – விளையாடுதல்தான் ஆக்கப்பூர்வமான வேலைக்கு மிகவும் திறமையான முறை

மாறாக, மென்பொருளைப் பற்றிய ஒரு பரந்த பாடம் என்ன என்பதை நான் பரிந்துரைக்க விரும்புகிறேன் (மற்றும் அநேகமாக ஒவ்வொரு வகையான ஆக்கப்பூர்வமான அல்லது தொழில்முறை வேலைகளையும் பற்றி). மனிதர்கள் பொதுவாக உகந்த-சவால் மண்டலத்தில் (optimal-challenge zone) ஒரு பணி விழும்போது அதில் மகிழ்ச்சி அடைகிறார்கள். மிகவும் எளிதானது என்றால் சலிப்பை ஏற்படுத்தும். செய்து முடிக்க மிகவும் கடினமாகவும் இருக்கக் கூடாது. சவாலற்ற வேலையோ அல்லது தவறாக வடிவமைக்கப்பட்ட இலக்குகளோ அல்லது அழுத்தமான செயல்முறை பாரங்களோ இல்லாவிட்டால் அவர் ஒரு மகிழ்ச்சியான நிரலாளர். வேலை செய்யும்போது மகிழ்வூட்டும் உணர்வு நல்ல செயல்திறனை முன்னறிவிக்கிறது.

பயம் மற்றும் வெறுப்புடன் உங்களின் சொந்த வேலைச் செயல்முறையைத் தொடர்புபடுத்துவது (தில்பர்ட் கார்ட்டூன்களைத் தொங்கவிடுவதன் மூலம் பரிந்துரைக்கப்படும் முரண்பாடான வழியில் கூட) செயல்முறை தோல்வியடைந்ததற்கான அறிகுறியாகவே கருதப்பட வேண்டும். மகிழ்ச்சி, நகைச்சுவை மற்றும் விளையாட்டுத்தனம் ஆகியவை உண்மையில் சொத்துக்கள். மேலே உள்ள

“மகிழ்ச்சியான கூட்டங்கள் (happy hordes)” பற்றி நான் எழுதுவது எதுகை மோனைக்காக அல்ல. மேலும் லினக்ஸ் சின்னம் ஒரு அரவணைக்கத் தூண்டும் குழந்தைமுகப் பென்குயின் (penguin) என்பது வெறும் நகைச்சுவையல்ல.

திறந்த மூல வெற்றியின் மிக முக்கியமான விளைவுகளில் ஒன்று, ஆக்கப்பூர்வமான வேலைக்கு பொருளாதார ரீதியாக மிகவும் திறமையான முறை விளையாடுதல்தான் என்பதை நமக்குக் கற்பிப்பதேயாகும்.



திறந்த மூல ஃபயர்ஃபாக்ஸ் உலாவி வெளியிடும் மோசில்லா அறக்கட்டளையின் அலுவலகம் **Kaniyam Foundation Account Number : 606 1010 100 502 79 Union Bank Of India West Tambaram, Chennai IFSC – UBIN0560618 Account Type : Current Account**

முடிவுரை: சந்தை பாணியை நெட்ஸ்கேப் தழுவுகிறது

நெட்ஸ்கேப் **(Netscape)** நிறுவனம் அதன் கம்யூனிகேட்டர் **(Communicator)** உலாவியைத் திறந்த மூலமாக வெளியிடும் திட்டத்தை அறிவித்தது

நீங்கள் வரலாறு படைக்க உதவுகிறீர்கள் என்பது ஒரு விசித்திரமான உணர்வு....

ஜனவரி 22, 1998 இல், நான் முதன்முதலில் பேராலயமும் சந்தையும் வெளியிட்ட சுமார் ஏழு மாதங்களுக்குப் பிறகு, நெட்ஸ்கேப் (Netscape) நிறுவனம் அதன் கம்யூனிகேட்டர் (Communicator) உலாவியின் மூடிய மூலமாக இருந்த நிரலைத் திறந்த மூலமாக வெளியிடும் திட்டத்தை அறிவித்தது. அறிவிப்பு வரும் நாளுக்கு முன்பு இது நடக்கப் போகிறது என்று எனக்கு எதுவும் தெரியாது.

நெட்ஸ்கேப்பின் நிர்வாகத் துணைத் தலைவரும், தலைமை தொழில்நுட்ப அதிகாரியுமான எரிக் ஹான் (Eric Hahn), அதற்குப் பிறகு எனக்கு மின்னஞ்சல் அனுப்பினார்: “நெட்ஸ்கேப்பில் உள்ள அனைவரின் சார்பாக, நாங்கள் இந்த முடிவுக்கு வர உதவியதற்கு முதலில் நன்றி தெரிவிக்க விரும்புகிறேன். உங்களின் சிந்தனையும் எழுத்துக்களும் எங்கள் முடிவுக்கு அடிப்படை

உத்வேகமாக இருந்தன.”

இதற்கு அடுத்த வாரம், நெட்ஸ்கேப்பின் அழைப்பின் பேரில் நான் சிலிக்கான் பள்ளத்தாக்குக்கு (Silicon Valley) ஒரு நாள் முழுவதும் (4 பிப்ரவரி 1998 அன்று) அவர்களின் சில முக்கிய நிர்வாகிகள் மற்றும் தொழில்நுட்ப நபர்களுடன் உத்தி மாநாட்டிற்குச் சென்றேன். நெட்ஸ்கேப்பின் மூல-வெளியீட்டு உத்தியையும் உரிமத்தையும் ஒன்றாக வடிவமைத்தோம்.

வணிக உலகில் சந்தை மாதிரியின் பெரிய அளவிலான, நிஜ உலக சோதனையை நெட்ஸ்கேப் நமக்கு வழங்க உள்ளது

சில நாட்களுக்குப் பிறகு நான் பின்வருமாறு எழுதினேன்:

வணிக உலகில் சந்தை மாதிரியின் பெரிய அளவிலான, நிஜ உலக சோதனையை நெட்ஸ்கேப் நமக்கு வழங்க உள்ளது. திறந்த மூல கலாச்சாரம் இப்போது ஆபத்தை எதிர்கொள்கிறது. நெட்ஸ்கேப்பின் செயல்படுத்தல் வேலை செய்யவில்லை என்றால், திறந்த மூலக் கருத்து மிகவும் மதிப்பிழந்து, வணிக உலகம் இன்னும் ஒரு பத்தாண்டுகளுக்கு அதைத் தொடாது.

மறுபுறம், இது ஓர் அற்புதமான வாய்ப்பு. அமெரிக்க பங்கு சந்தை மற்றும் பிற இடங்களில் இந்த நடவடிக்கையை எச்சரிக்கையுடன் ஆனால் நல்லவிதமாகப் பார்க்கிறார்கள். நம்மை நிரூபிக்க நமக்கும் ஒரு வாய்ப்பு கொடுக்கப்படுகிறது. இந்த நடவடிக்கையின் மூலம் நெட்ஸ்கேப் கணிசமான சந்தைப் பங்கை மீண்டும் பெற்றால், அது மென்பொருள் துறையில் நீண்ட காலமாக எதிர்பார்த்த புரட்சியை ஏற்படுத்தலாம்.

அடுத்த ஆண்டு மிகவும் கற்கக்கூடிய மற்றும் சுவாரசியமான நேரமாக இருக்க வேண்டும்.

மைக்ரோசாப்ட் உலாவி சந்தையில் ஏகபோக நிலையை அடைவதை மோசில்லா ஃபயர்ஃபாக்ஸ் **(Mozilla Firefox)** உலாவி தடுத்தது

உண்மையில் அவ்வாறே அது இருந்தது. 2000 ஆம் ஆண்டின் நடுப்பகுதியில் நான் எழுதிக்கொண்டிருக்கும் நேரத்தில், பின்னர் மோசில்லா (Mozilla) என்று பெயரிடப்பட்ட அதன் வளர்ச்சி ஒரு நிபந்தனைக்குட்பட்ட வெற்றியை மட்டுமே அடைந்தது. இது நெட்ஸ்கேப்பின் அசல் இலக்கை அடைந்தது. அதாவது இது மைக்ரோசாப்ட் உலாவி சந்தையில் ஏகபோக நிலையை அடைவதைத் தடுப்பது. இது சில வியத்தகு வெற்றிகளையும் அடைந்துள்ளது (குறிப்பாக அடுத்த தலைமுறை கெக்கோ வரைவிக்கும் பொறியின் (Gecko rendering engine) வெளியீடு).

இருப்பினும், மோசில்லா நிறுவனர்கள் முதலில் எதிர்பார்த்த அளவு நெட்ஸ்கேப்பிற்கு வெளியில் இருந்து பெரிய அளவில் நிரலாளர்களின் பங்களிப்பை இது இன்னும் பெறவில்லை. இங்குள்ள பிரச்சினை என்னவென்றால், நீண்ட காலமாக மோசில்லா விநியோகம் உண்மையில் சந்தை மாதிரியின் அடிப்படை விதிகளில் ஒன்றை உடைத்துவிட்டது. பங்களிக்க விரும்புபவர்கள் எளிதில் ஓட்டி வேலை செய்வதைப் பார்க்கக்கூடிய ஒன்றை அது அனுப்பவில்லை. (மேலும், வெளியிடப்பட்ட ஒரு வருடத்திற்கும் மேலாகும் வரை, மூல நிரலிலிருந்து மோசில்லாவை இருமமாக்க (compile), தனியுரிம மோட்டிஃப் (Motif) நிரலகத்திற்கான உரிமம் தேவைப்பட்டது.)

“திறந்த மூலம் ஒரு மந்திரப் பொடி அல்ல”

முக்கியப் பிரச்சினையாக (வெளியுலகின் பார்வையில்) மோசில்லா குழுமம் திட்டத் துவக்கத்திற்குப் பிறகு இரண்டரை ஆண்டுகளுக்கு ஓர் உற்பத்தித்தரமான உலாவியை வெளியிடவில்லை. மேலும் 1999 இல் திட்டத்தின் முதன்மையாளர்களில் ஒருவர் தனது வேலையைத் துறப்பதன் மூலம் சற்றுப் பரபரப்பை ஏற்படுத்தினார். மோசமான நிர்வாகம் மற்றும் தவறவிட்ட வாய்ப்புகள் குறித்து புகார் செய்தார். “திறந்த மூலம்,” அவர் சரியாகக் கூறினார், “ஒரு மந்திரப் பொடி அல்ல”.

உண்மையில் திறந்த மூலம் மந்திரப் பொடி அல்ல. ஜேமி ஜாவின்ஸ்கியின் (Jamie Zawinski) ராஜினாமா கடிதத்தின் போது இருந்ததை விட, மோசில்லாவின் நீண்ட கால முன்கணிப்பு இப்போது (நவம்பர் 2000 இல்) சிறப்பாகத் தெரிகிறது. கடந்த சில வாரங்களில் அன்றாட வெளியீடுகள் இறுதியாக உற்பத்திப் பயன்பாட்டிற்கான முக்கியமான வரம்பைக் கடந்துள்ளன. ஆனால் சரியாக வரையறுக்கப்படாத இலக்குகள், குழப்பமான நிரல் அல்லது மென்பொருள் பொறியியலின் பிற நாட்பட்ட பிரச்சினைகளால் பாதிக்கப்படும் ஏற்கனவே இருக்கும் திட்டத்தைக் காப்பாற்ற முடியாது என்று ஜேமி சரியாகச் சுட்டிக்காட்டினார். திறந்த மூலம் எவ்வாறு வெற்றிபெறும் மற்றும் அது எவ்வாறு தோல்வியடையும் என்ற இரண்டுக்கும் ஒரே நேரத்தில் மோசில்லா எடுத்துக்காட்டாகியுள்ளது.

இருப்பினும், இதற்கிடையில், திறந்த மூல யோசனை வெற்றிகளைப் பெற்றுள்ளது மற்றும் பிற இடங்களில் ஆதரவாளர்களைக் கண்டறிந்துள்ளது. நெட்ஸ்கேப் வெளியீட்டிற்குப் பிறகு, திறந்த மூல மேம்பாட்டு மாதிரியில் ஆர்வத்தின் மிகப்பெரிய அதிகரிப்பைக் கண்டோம். இந்தப் போக்கு லினக்ஸ் இயங்குதளத்தின் தொடர்ச்சியான வெற்றியால் இயக்கப்படுகிறது மற்றும் அதையும் உந்துகிறது.

மோசில்லா எட்டிய வெற்றிப் போக்கு வேகமான விகிதத்தில் தொடர்கிறது.

கணியம் அறக்கட்டளை



தொலை நோக்கு – **Vision**

தமிழ் மொழி மற்றும் இனக்குழுக்கள் சார்ந்த மெய்நிகர்வளங்கள், கருவிகள் மற்றும் அறிவுத்தொகுதிகள், அனைவருக்கும் கட்டற்ற அணுக்கத்தில் கிடைக்கும் சூழலை உருவாக்குதல்.

பணி இலக்கு – **Mission**

அறிவியல் மற்றும் சமூகப் பொருளாதார வளர்ச்சிக்கு ஒப்ப, தமிழ் மொழியின் பயன்பாடு வளர்வதை உறுதிப்படுத்துவதும், அனைத்து அறிவுத் தொகுதிகளும், வளங்களும் கட்டற்ற அணுக்கத்தில் அனைவருக்கும் கிடைக்கச்செய்தலும்.

எமது பணிகள்

- கணியம் மின்னிதழ் - kaniyam.com
- கணிப்பொறி சார்ந்த கட்டுரைகள், காணொளிகள், மின்னூல்களை இங்கு வெளியிடுகிறோம்.
- கட்டற்ற தமிழ் நூல்கள் - FreeTamilEbooks.com
- இங்கு யாவரும் எங்கும் பகிரும் வகையில், கிரியேட்டிவ் காமன்ஸ் உரிமையில், தமிழ் மின்னூல்களை இலவசமாக, அனைத்துக்

கருவிகளிலும் படிக்கும் வகையில் epub, mobi, A4 PDF, 6 inch PDF வடிவங்களில் வெளியிடுகிறோம்.

- தமிழுக்கான கட்டற்ற மென்பொருட்கள் உருவாக்கம்
- தமிழ் ஒலியோடைகள் உருவாக்கி வெளியிடுதல்
- விக்கி மூலத்தில் உள்ள மின்னூல்களை பகுதிநேர/முழு நேரப் பணியாளர்கள் மூலம் விரைந்து பிழை திருத்துதல்
- OpenStreetMap.org ல் உள்ள இடம், தெரு, ஊர் பெயர்களை தமிழாக்கம் செய்தல்.

மேற்கண்ட திட்டங்கள், மென்பொருட்களை உருவாக்கி செயல்படுத்த உங்கள் அனைவரின் ஆதரவும் தேவை. உங்களால் எவ்வாறேனும் பங்களிக்க இயலும் எனில் உங்கள் விவரங்களை kaniyamfoundation@gmail.com க்கு மின்னஞ்சல் அனுப்புங்கள்.

வெளிப்படைத்தன்மை

கணியம் அறக்கட்டளையின் செயல்கள், திட்டங்கள், மென்பொருட்கள் யாவும் அனைவருக்கும் பொதுவானதாகவும், முழுமையான வெளிப்படைத்தன்மையுடனும் இருக்கும். https://github.com/KaniyamFoundation/Organization/issues இந்த இணைப்பில் செயல்களையும், https://github.com/KaniyamFoundation/Organization/wiki இந்த இணைப்பில் மாத அறிக்கை, வரவு செலவு விவரங்களுடனும் காணலாம்.

கணியம் அறக்கட்டளையில் உருவாக்கப்படும் மென்பொருட்கள் யாவும் கட்டற்ற மென்பொருட்களாக மூல நிரலுடன், GNU GPL, Apache, BSD, MIT, Mozilla ஆகிய உரிமைகளில் ஒன்றாக

வெளியிடப்படும். உருவாக்கப்படும் பிற வளங்கள், புகைப்படங்கள், ஒலிக்கோப்புகள், காணொளிகள், மின்னூல்கள், கட்டுரைகள் யாவும் யாவரும் பகிரும், பயன்படுத்தும் வகையில் கிரியேட்டிவ் காமன்சு உரிமையில் இருக்கும்.

நன்கொடை

உங்கள் நன்கொடைகள் தமிழுக்கான கட்டற்ற வளங்களை உருவாக்கும் செயல்களை சிறந்த வகையில் விரைந்து செய்ய ஊக்குவிக்கும்.

பின்வரும் வங்கிக் கணக்கில் உங்கள் நன்கொடைகளை அனுப்பி, உடனே விவரங்களை kaniyamfoundation@gmail.com க்கு மின்னஞ்சல் அனுப்புங்கள்.